



NOMS 2014 Tutorial Sessions

TUTORIAL T3

Monday, May 5, 2014

Configuration, Management, and Enablement in a Software Defined Platform

Salman A. Baset, IBM Research, U.S.A.

Theo Benson, Duke University, U.S.A.

Sambit Sahu, IBM Research, U.S.A.

NOMS 2014 Tutorial Co-chairs:

Mehmet Ulema, Manhattan College, U.S.A.

Burkhard Stiller, University of Zürich, Switzerland

Motivation: Cloud in Transition

- First wave of Cloud
 - Amazon, Rackspace
 - Focused on CapEx reduction
- Second Wave of Cloud
 - DevOps, Agility
 - Software Defined X

Software Defined Platforms

- Part 1: Cloud Configuration and Management
- Part 2: Cloud Platform Enablement (OpenStack, AWS)
- Part 3: Cloud Network Enablement: Software Defined Networks

Cloud On-demand Resource Provisioning Basics

Amazon AWS console (EC2 view)

The screenshot displays the Amazon AWS console interface, specifically the Amazon EC2 Console Dashboard. The top navigation bar includes the AWS logo, the user's name (sambit sahu), and links for Settings and Sign Out. Below the navigation bar, a horizontal menu lists various AWS services, with EC2 highlighted. The left sidebar contains a navigation menu with categories like INSTANCES, IMAGES, ELASTIC BLOCK STORE, and NETWORKING & SECURITY, each with sub-links. The main content area is titled 'Amazon EC2 Console Dashboard' and is divided into three sections: 'Getting Started', 'My Resources', and 'Service Health'. The 'Getting Started' section provides instructions on how to launch a virtual server and includes a 'Launch Instance' button. The 'My Resources' section shows a summary of the user's EC2 resources in the US East (Virginia) region, including 1 Running Instance, 0 Elastic IPs, 1 EBS Volume, 0 EBS Snapshots, 2 Key Pairs, 6 Security Groups, 0 Load Balancers, and 0 Placement Groups. The 'Service Health' section displays the current status of the Amazon EC2 service, which is 'Service is operating normally'.

aws.amazon.com | AWS | Products | Developers | Community | Support | Account | Welcome, sambit sahu | Settings | Sign Out

AWS Elastic Beanstalk Amazon S3 Amazon EC2 Amazon VPC Amazon CloudWatch Amazon Elastic MapReduce Amazon CloudFront Amazon RDS Amazon SNS

Navigation
Region: US East

> EC2 Dashboard

INSTANCES

- > Instances
- > Spot Requests

IMAGES

- > AMIs
- > Bundle Tasks

ELASTIC BLOCK STORE

- > Volumes
- > Snapshots

NETWORKING & SECURITY

- > Elastic IPs
- > Security Groups
- > Placement Groups
- > Load Balancers
- > Key Pairs

Amazon EC2 Console Dashboard

Getting Started

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

[Launch Instance](#)

Note: Your instances will launch in the US East (Virginia) region.

My Resources

You are using the following Amazon EC2 resources in the US East (Virginia) region:

- 1 Running Instance
- 0 Elastic IPs
- 1 EBS Volume
- 0 EBS Snapshots
- 2 Key Pairs
- 6 Security Groups
- 0 Load Balancers
- 0 Placement Groups

Service Health

Current Status	Details
✓	Amazon EC2 (US East - N. Virginia) Service is operating normally

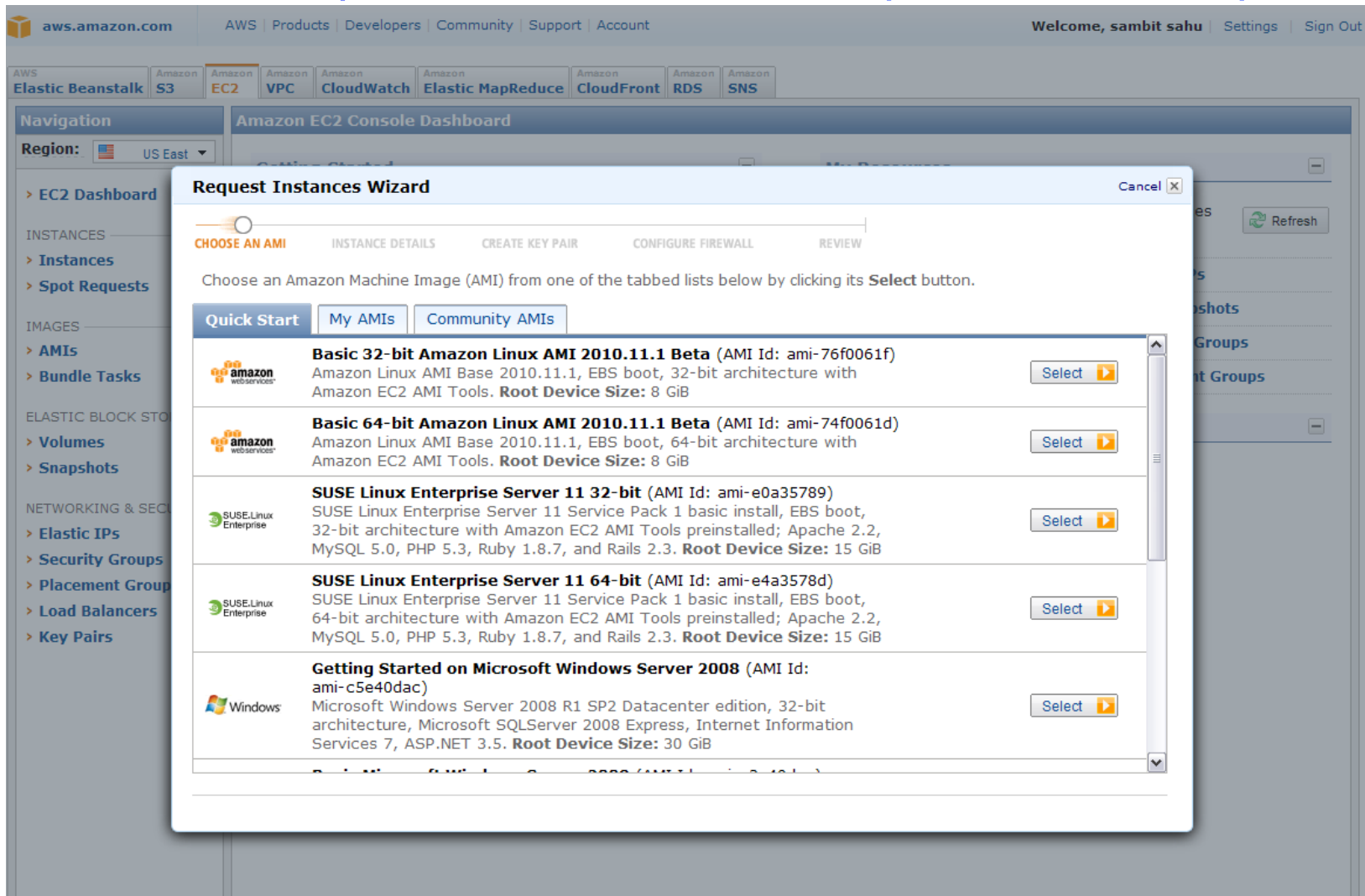
[View complete service health details](#)

Related Links

- > Documentation
- > All EC2 Resources
- > Forums
- > Feedback
- > Report an Issue

- User logs in with AWS credentials

User launches request instance → a list of prebuilt stack is provided



- AWS shows a list of available pre-built base software stack (called **Virtual Appliances**) user may request to add to the machine

User can choose the resource size (CPU, mem choices)

The screenshot shows the AWS Management Console with the 'Request Instances Wizard' dialog box open. The wizard is in the 'INSTANCE DETAILS' step. The 'Number of Instances' is set to 1, and the 'Availability Zone' is set to 'No Preference'. The 'Instance Type' dropdown is open, showing a table of available instance types.

Type	CPU Units	CPU Cores	Memory
Micro (t1.micro)	Up to 2 ECUs	1 Core	613 MB
Small (m1.small)	1 ECU	1 Core	1.7 GB
High-CPU Medium (c1.medium)	5 ECUs	2 Cores	1.7 GB

- Instance request wizard guides through resource choices

User specifies security/access configurations

The screenshot displays the AWS Management Console interface. At the top, the navigation bar includes the AWS logo, the URL 'aws.amazon.com', and links for 'AWS | Products | Developers | Community | Support | Account'. The user is logged in as 'Welcome, sambit sahu' with links for 'Settings' and 'Sign Out'. Below the navigation bar, a horizontal menu lists various AWS services: Elastic Beanstalk, S3, EC2, VPC, CloudWatch, Elastic MapReduce, CloudFront, CloudFormation, RDS, SNS, and IAM. The 'EC2' service is selected, leading to the 'Amazon EC2 Console Dashboard'. The dashboard has two tabs: 'Getting Started' and 'My Resources'. A 'Request Instances Wizard' dialog box is open in the foreground. The wizard has a progress bar with five steps: 'CHOOSE AN AMI', 'INSTANCE DETAILS', 'CREATE KEY PAIR' (the current step), 'CONFIGURE FIREWALL', and 'REVIEW'. The 'CREATE KEY PAIR' step contains the following text: 'Public/private key pairs allow you to securely connect to your instance after it launches. To create a key pair, enter a name and click **Create & Download your Key Pair**. You will then be prompted to save the private key to your computer. Note, you only need to generate a key pair once - not each time you want to deploy an Amazon EC2 instance.' Below this text are three radio button options: 'Choose from your existing Key Pairs' (selected), 'Create a new Key Pair', and 'Proceed without a Key Pair'. The 'Choose from your existing Key Pairs' option has a dropdown menu showing 'IM2011'. At the bottom of the wizard, there are '< Back' and 'Continue >' buttons.

aws.amazon.com | AWS | Products | Developers | Community | Support | Account | Welcome, sambit sahu | Settings | Sign Out

AWS Elastic Beanstalk S3 EC2 VPC CloudWatch Elastic MapReduce CloudFront CloudFormation RDS SNS IAM

Navigation
Region: US East (Virginia)
EC2 Dashboard
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORKING & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Key Pairs

Amazon EC2 Console Dashboard
Getting Started | My Resources

Request Instances Wizard [Cancel]

CHOOSE AN AMI | INSTANCE DETAILS | **CREATE KEY PAIR** | CONFIGURE FIREWALL | REVIEW

Public/private key pairs allow you to securely connect to your instance after it launches. To create a key pair, enter a name and click **Create & Download your Key Pair**. You will then be prompted to save the private key to your computer. Note, you only need to generate a key pair once - not each time you want to deploy an Amazon EC2 instance.

☒ Choose from your existing Key Pairs

Your existing Key Pairs*: IM2011

☐ Create a new Key Pair

☐ Proceed without a Key Pair

< Back | Continue >

AWS provisions an instance and returns user credentials

AWS

Elastic Beanstalk

Amazon

S3

Amazon

EC2

Amazon

VPC

Amazon

CloudWatch

Amazon

Elastic MapReduce

Amazon

CloudFront

AWS

CloudFormation

Amazon

RDS

Amazon

SNS

AWS

IAM

Navigation

Region:

US East (Virginia)

> EC2 Dashboard

INSTANCES

> Instances

> Spot Requests

> Reserved Instances

IMAGES

> AMIs

> Bundle Tasks

ELASTIC BLOCK STORE

> Volumes

> Snapshots

NETWORKING & SECURITY

> Security Groups

> Elastic IPs

> Placement Groups

> Load Balancers

> Key Pairs

My Instances

Launch Instance

Instance Actions

Show/Hide

Refresh

Help

Viewing: All Instances All Instance Types 1 to 3 of 3 Instances

	Name	Instance	AMI ID	Root Device	Type	Status	Security Groups	Key Pair Name	Monitoring	Virtualization	Placement
☐	empty	i-a1c318cf	ami-e4a3578d	ebs	t1.micro		IM2001	IM2011	basic	paravirtual	
☐	MyFirstInstance	i-3b7aa155	ami-76f0061f	ebs	m1.small		default		basic	paravirtual	
☒		i-6176ad0f	ami-e4a3578d	ebs	t1.micro		IM2001	IM2011		paravirtual	

1 EC2 Instance selected

EC2 Instance: i-6176ad0f

Description

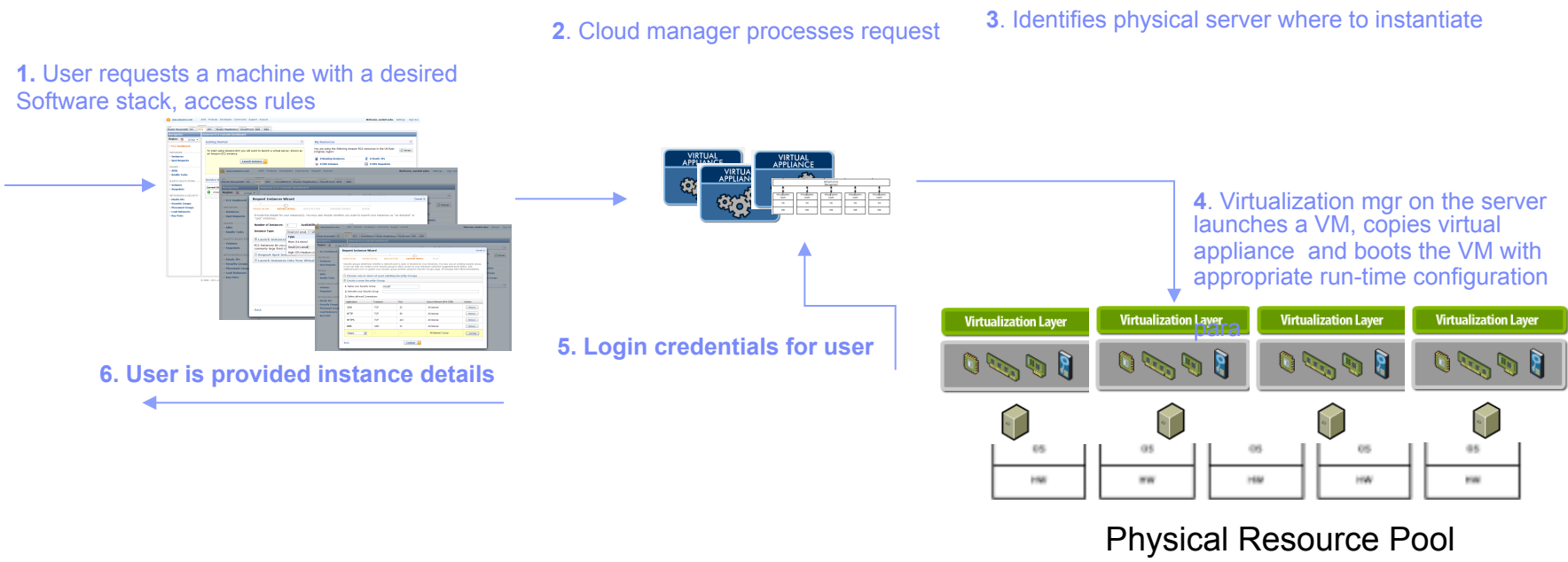
Monitoring

Tags

AMI:	sles-11-sp1-v1.00.x86_64 (ami-e4a3578d)	Zone:	us-east-1c
Security Groups:	IM2001	Type:	t1.micro
Status:	running	Owner:	026317314573
VPC ID:	-	Subnet ID:	-
Source/Dest. Check:		Virtualization:	paravirtual
Placement Group:		Reservation:	r-ddfe6db1
RAM Disk ID:	-	Platform:	-
Key Pair Name:	IM2011	Kernel ID:	aki-427d952b
Monitoring:	detailed	AMI Launch Index:	0
Elastic IP:	-	Root Device:	sda1
Root Device Type:	ebs	Tenancy:	default
Lifecycle:	normal		
Block Devices:	sda1		
Public DNS:	ec2-50-16-69-93.compute-1.amazonaws.com		
Private DNS:	ip-10-196-229-93.ec2.internal		
Private IP Address:	10.196.229.93		
Launch Time:	2011-05-26 10:45 EDT		
State Transition Reason:			

Deconstructing Amazon EC2 request machine API

- User goes to Amazon EC2 portal and specifies desired parameters for a machine
 - Resource: CPU, mem, disk
 - Stack: OS and possibly with additional software
- Amazon AWS Cloud manager (resource pool manager) provisions the user request
 - Finds appropriate physical resource
 - Dispatches the request to virtualization manager on the identified resource
 - Cloud Manager invokes EC2 API to provisions the request
- Virtualization manager on physical server
 - Copies the pre-built software stack (virtual appliance)
 - Provisions a guest VM and configures parameters (IP address, access rules,...) at run/boot time
- Cloud manager returns login credentials to user



Part 1. Configuration and Management

Cloud Configuration and Management

- Cloud DevOps
 - What is DevOps?
 - Overview of DevOps and challenges
 - Tools: Chef and Puppet
 - Chef example
- Amazon OpsWorks

What is DevOps?

- Processes to help rapidly **produce**, **deploy**, **operate**, and **upgrade** software, typically, for large scale systems, through standardization and automation, while meeting requirements such as:
 - Zero-customer down time
 - No SLA violation

DevOps cycle: produce, deploy, operate, upgrade

- Produce software
 - Develop software in a standardized way
 - What does it mean?
 - Traditionally,
 - Write code
 - perform simple verification test (SVT)
 - Perform functional verification test (FVT)
 - Integrate code
 - Perform integration testing
 - Release alpha
 - Release beta
 - Release software on a DVD
 - Why this model does not work well in the era of cloud?
 - Too slow
 - Software release cycle measured in Months and Years
 - Do you know how many updates to Facebook per month?
 - Two per day

<https://www.facebook.com/notes/facebook-engineering/ship-early-and-ship-twice-as-often/10150985860363920>

DevOps cycle: produce, deploy, operate, upgrade

- OpenStack development
- Uses git, jenkins, Gerrit
- Each commit goes through a series of tests
- If any of the tests fail, a commit cannot be accepted
- Then, a commit must be reviewed by two core developers
- Then, it is merged into mainline OpenStack
- Typically, new code must be accompanied by test code

<https://review.openstack.org/#/q/status:open,n,z>

https://wiki.openstack.org/wiki/How_To_Contribute

DevOps cycle: produce, deploy, operate, upgrade

- Deploy Software
 - Traditionally
 - Get a software DVD, Install it, Configure it
- Why this model does not work well in the era of cloud?
 - Modern distributed systems are complex
 - OpenStack has 5-7 controller machines, 100s of compute nodes
 - > 40 configuration files
 - Netflix is deployed on more than 10K machines, according to one estimate
 - Can't deploy it manually
 - Chasm between developers and operators/deployers
 - Developers focus on functionality, they do not focus on how to deploy
 - Deployers/operators have no insight into how things are changing in a release
 - Dev --- Ops

Treat infrastructure as code!!!

DevOps cycle: produce, deploy, operate, upgrade

- Operate software
- Challenges
 - Multiple tools, e.g.,
 - Monitoring tool, log tool, Provisioning dashboard
 - Multiple UIs

DevOps cycle: produce, deploy, operate, upgrade

- Upgrade software
- Zero-down time upgrade
- Challenges
 - Do all software support upgrade?
 - How does Facebook perform upgrades on its site and infrastructure?

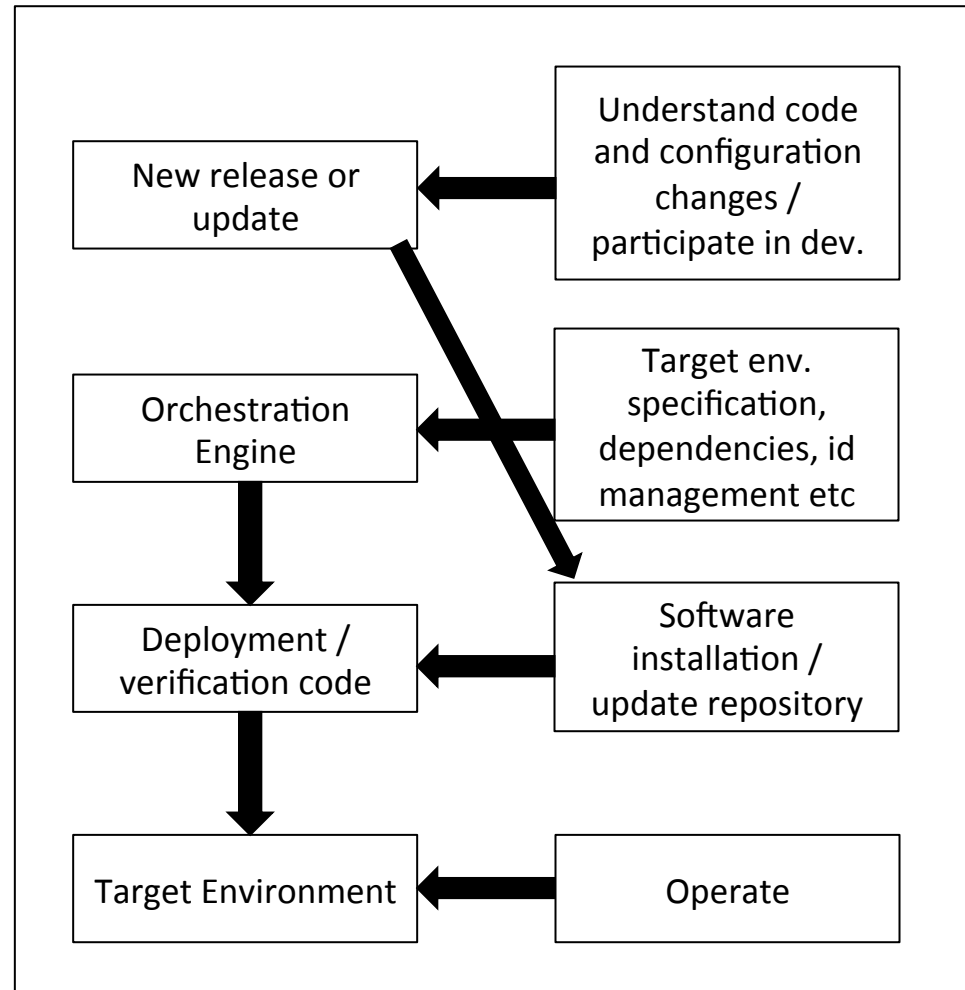
Summary of DevOps challenges for large scale distributed/cloud systems

- Complexity
 - OpenStack, 1-10 controller machines, 100 compute nodes
 - ITM, 1-10 controller machines, thousands of ITM agents
- Not all software is home grown
 - E.g., OpenStack, Cloud Foundry
- Frequent releases
 - E.g., OpenStack release every six months
 - Consuming directly from dev. trunk
- Thousands of configuration options
 - E.g., > 2000 parameters for OpenStack, ITM
- Multiplatform
 - E.g, RHEL, Ubuntu

DevOps process (intermediate detail)

User or customer

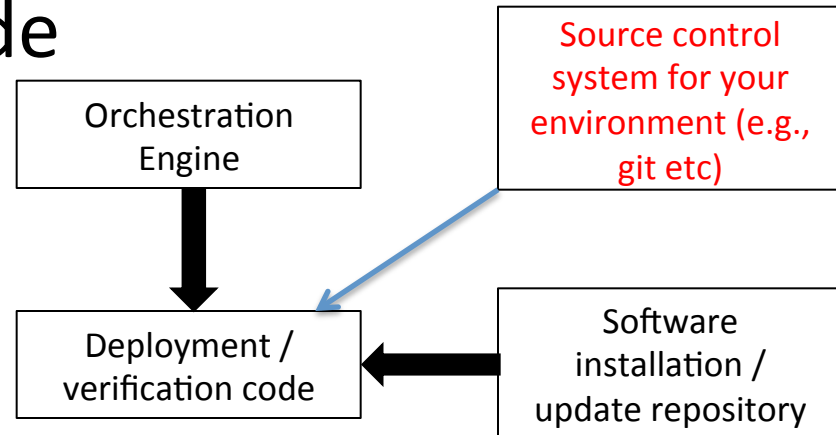
- Where is deployment code hosted?
 - In a source control system, e.g., git
- Where is software installation repo?
 - Local to the environment, over the network, or Internet



Cloud DevOps cycle: produce, DEPLOY, operate, upgrade

- Deploying complex distributed software

- Can't do manually
- How to deploy?
- Treat infrastructure as code



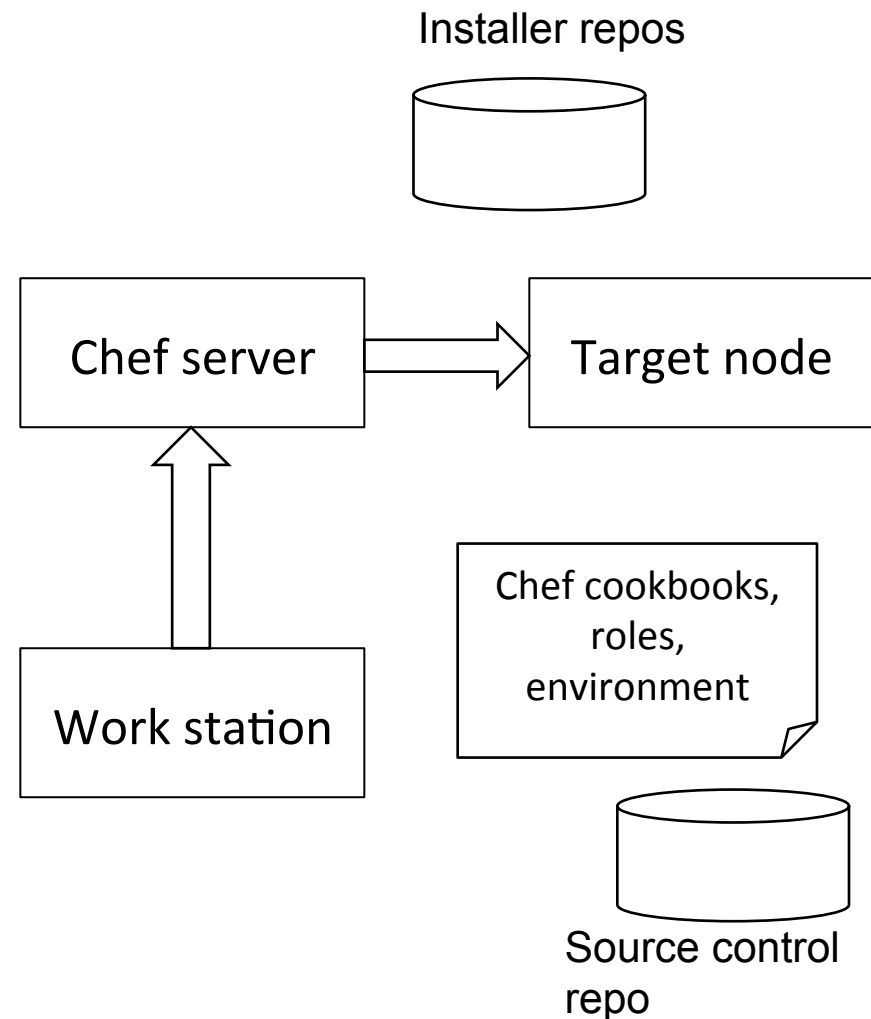
- (1) Develop deployment code in Chef or Puppet OR
Leverage existing deployment code from open source community
- (2) Update deployment code into your source control system
- (3) Update software installation repository
- (4) Run the deployment code through an orchestration engine
- (5) Verify deployment. Revert back, if possible

Challenges with automated deployment?

- How do I know that automatic deployment worked?
- How do I know that configurations are correct?

Chef

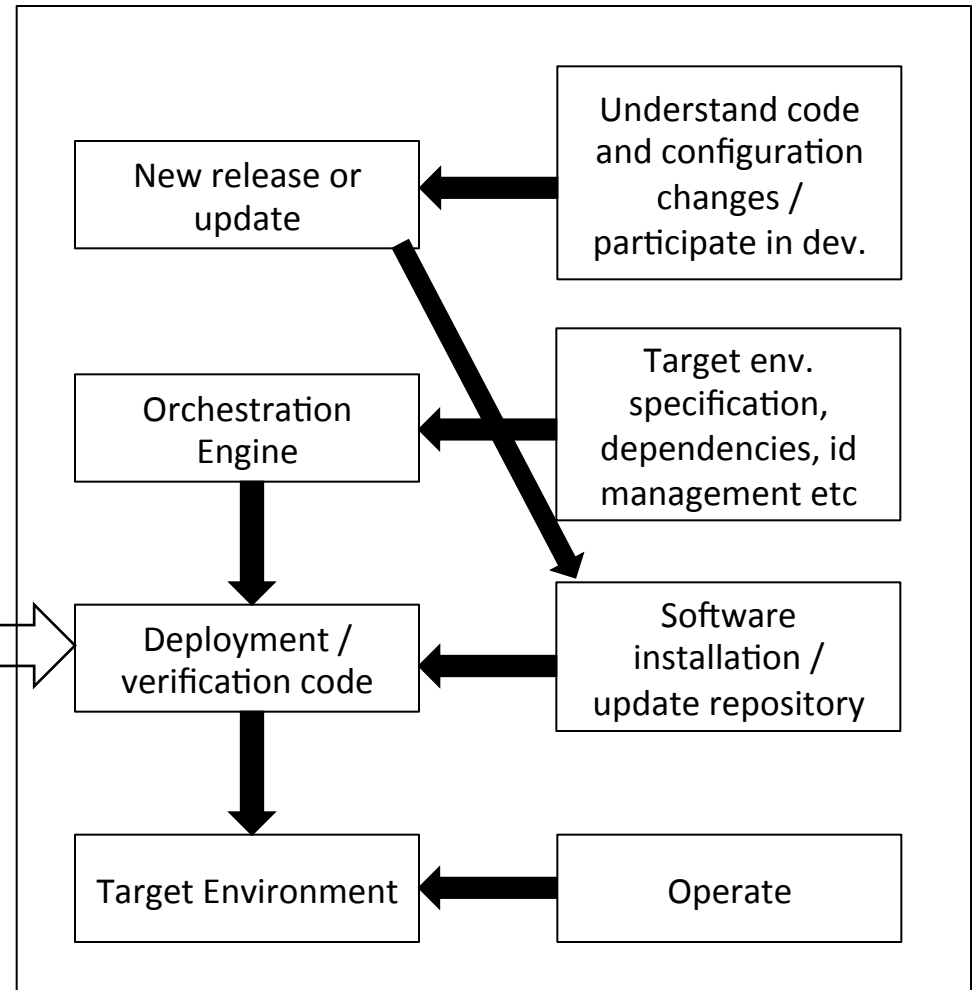
- What is Chef?
 - A configuration management and deployment tool written in Ruby and Erlang
- Chef components
 - Server, workstation, node
 - Workstation contains Chef deployment code hosted in a source control repository.
 - The code is loaded in Chef server
 - External components
 - Package repositories
 - Chef server not always needed
 - Chef solo



What Chef (or Puppet) can or cannot do?

- Chef is not an orchestration engine
- Chef does not contain installers or packages to be installed.

**Chef /
Puppet**



Chef deployment code

- A cookbook
- Each cookbook has a bunch of recipes (deployment code)
- Recipes are invoked by roles defined per component or sub-component
 - E.g., role[os-database]
 - role[os-compute-controller-api]
 - role[os-image-api], role[os-network-controller-api],
 - Role is typically a noun.
- Recipes can deploy according to environments (e.g., RHEL, Ubuntu), architectures, (32-bit, 64-bit)
- Multiple roles can be applied to the same node

Structure of a **cookbook**

attributes/	←	An attribute typically defines a setting on a node.
CHANGELOG.md		
recipes/	←	A piece of code to configure a node fully or partly.
definitions/		
Files		A resource defines the desired state for a single configuration item present on a node that is under management by Chef (function declaration)
resources/	←	
README.md		
providers/	←	A provider defines the steps that are needed to bring that piece of the system from its current state into the desired state (function definition).
libraries/		
metadata.rb		
templates/	←	A cookbook template is an Embedded Ruby (ERB) template that is used to generate files based on the variables and logic contained within the template. Useful in creating configuration files.

MySQL cookbook

- Let's discuss it in the context of MySQL cookbook
 - <https://github.com/opscode-cookbooks/mysql>

Structure of a **mysql cookbook** - attributes

- `default['mysql']['service_name'] = 'default'`
- `passwords default['mysql']['server_root_password'] = 'ilikerandompasswords'`
- `default['mysql']['server_debian_password'] = nil`
- `default['mysql']['server_repl_password'] = nil`

Passwords
used by MySQL

used in grants.sql

- `default['mysql']['allow_remote_root'] = false`
- `default['mysql']['remove_anonymous_users'] = true`
- `default['mysql']['root_network_acl'] = nil`
- **case** `node['platform']`
 - **when** `'smartos'`
 - `default['mysql']['data_dir'] = '/opt/local/lib/mysql'`
 - **else**
 - `default['mysql']['data_dir'] = '/var/lib/mysql'`
 - **End**
- `default['mysql']['port'] = '3306'`

Platform specific
attribute definition

Listening port
for mysql

Structure of a **mysql cookbook** - recipe

- **mysql_service** node['mysql']['service_name'] **do**
 - **port** node['mysql']['port']
 - **data_dir** node['mysql']['data_dir']
 - **server_root_password** node['mysql']['server_root_password']
 - **server_debian_password** node['mysql']['server_debian_password']
 - **server_repl_password** node['mysql']['server_repl_password']
 - **allow_remote_root** node['mysql']['allow_remote_root']
 - **remove_anonymous_users** node['mysql']['remove_anonymous_users']
 - **remove_test_database** node['mysql']['remove_test_database']
 - **root_network_acl** node['mysql']['root_network_acl']
 - action :create
- **end**

mysql_service resource – calls a provider

Structure of a **mysql** cookbook - resource

```
• require 'chef/resource/lwrp_base'
• require_relative 'helpers'

• extend Opscode::Mysql::Helpers
• #
• class Chef
•   class Resource
•     class MysqlService < Chef::Resource
•       extend Opscode::Mysql::Helpers
•       # Initialize resource
•       def initialize(name = nil, run_context = nil)
•         super
•         @resource_name = :mysql_service
•         @service_name = name

•         @allowed_actions = [:create, :restart, :reload]
•         @action = :create

•         # set default values
•         @version = default_version_for(
•           node['platform'],
•           node['platform_family'],
•           node['platform_version']
•         )

•         @package_name = package_name_for(
•           node['platform'],
•           node['platform_family'],
•           node['platform_version'],
•           @version
•         )

•         @data_dir = default_data_dir_for(node['platform_family'])
•         @port = '3306'
•         @template_source = nil

•         @allow_remote_root = false
•         @remove_anonymous_users = true
•         @remove_test_database = true
•         @root_network_acl = []

•         @server_root_password = 'ilikerandompasswords'
•         @server_debian_password = 'gnuslashlinux4ev4r'
•         @server_repl_password = nil
•       end
•     end
•   end
• end
```

Structure of a **mysql cookbook** - provider

- `require 'chef/provider/lwrp_base'`
- `require 'shellwords'`
- `require_relative 'helpers'`
- `extend Opscode::Mysql::Helpers`
- `class Chef`
- `class Provider`
- `class MysqlService`
- `class RHEL < Chef::Provider::MysqlService`
- `use_inline_resources if defined?(use_inline_resources)`
- `def whyrun_supported?`
- `true`
- `end`
- `end`
- `end`
- `end`
- `end`
- `action :create do`
- `case node['platform_version'].to_i.to_s`
- `when '2013'`
- `case new_resource.version`
- `when '5.1'`
- `base_dir = "`
- `include_dir = "#{base_dir}/etc/mysql/conf.d"`
- `prefix_dir = '/usr'`
- `lc_messages_dir = nil`
- `run_dir = '/var/run/mysqld'`
- `pid_file = '/var/run/mysql/mysql.pid'`
- `socket_file = '/var/lib/mysql/mysql.sock'`
- `package_name = 'mysql-server'`
- `service_name = 'mysqld'`
- `end`

Structure of a **mysql cookbook** - templates

- [mysqld] /etc/my.cnf
- user = mysql
- pid-file = <%= @pid_file %>
- socket = <%= @socket_file %>
- <% if @port %>
 - port = <%= @port %>
- <% end %>
- <% if @basedir %>
 - basedir = <%= @base_dir %>
- <% end %>
- <% if @data_dir %>
 - datadir = <%= @data_dir %>
- <% end %>

Chef roles

- A role is a way to define certain patterns and processes that exist across nodes in an organization as belonging to a single job function.
- Example – a role to deploy mysql. Calls mysql::server recipe
 - {
 - "name": "os-ops-database",
 - "description": "database deploy",
 - "json_class": "Chef::Role",
 - "default_attributes": {
 - },
 - "override_attributes": {
 - },
 - "chef_type": "role",
 - "run_list": [
 - "recipe[mysql::server]",
 -],
 - "env_run_lists": {
 - }
 - }

Deployment example

- `knife bootstrap 1.2.3.4 -x root -r role[os-ops-database] -d rhel -E NOMS14-env -N sql-node`

“DevOps as a service” – Amazon Opsworks



AWS OpsWorks

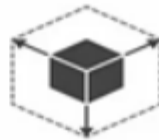
An integrated DevOps application management solution



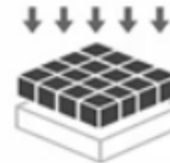
A **stack** represents the compute infrastructure and applications that you want to manage together.



A **layer** defines how to set up and configure a set of instances and related resources.



Decide how to scale: manually, with **24/7** instances, or automatically, with **load-based** or **time-based** instances.



Then deploy your app to specific instances and customize the deployment with Chef recipes.

Handles
life cycle events

- Various building blocks
 - Stacks
 - Layers
 - Instances
 - Apps
 - Customization

<https://aws.amazon.com/opsworks/getting-started/>

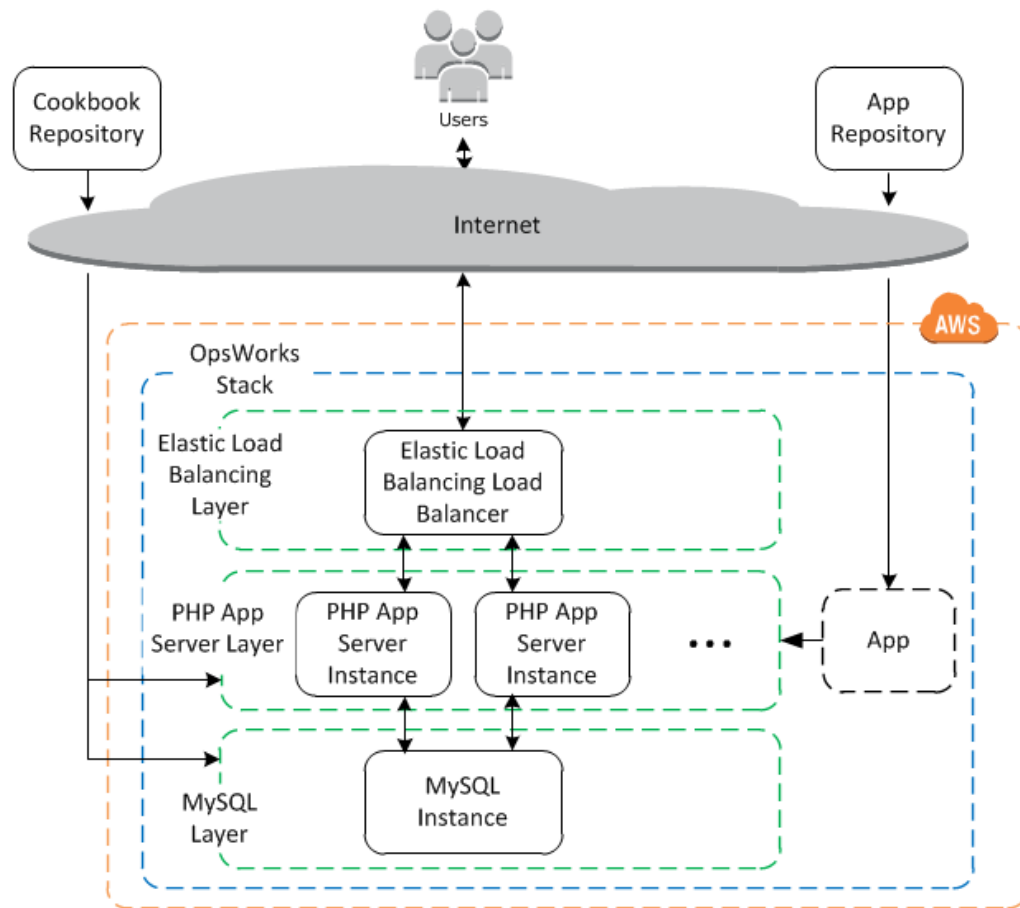
Amazon OpsWorks – Life Cycle Events (1/2)

- What are life cycle events?
- Setup
 - is sent to the instance when it is instantiated or successfully booted. For example, the OpsWorks Rails application server layer uses the setup event to trigger a Chef recipe that installs dependencies like Apache, Ruby, Passenger, and Ruby on Rails.
- Configure
 - notifies all instances whenever the state of the stack changes. For example, when a new instance is successfully added to an application server layer, the configure event triggers a Chef recipe that updates the OpsWorks Load Balancer layer configuration to reflect the added application server instance.

Amazon OpsWorks – Life Cycle Events (2/2)

- Deploy
 - is triggered whenever an application is deployed. For example, the OpsWorks Rails application server layer uses the deploy event to trigger a Chef recipe that executes the tasks needed to check out and download your application and tells Passenger to reload it.
- Undeploy
 - is sent when you delete an application. For example, the undeploy event can trigger a custom Chef recipe that specifies any cleanup steps that need to be run, such as deleting database tables.
- Shutdown
 - is sent to an instance 45 seconds before actually stopping the instance. For example, the shutdown event can trigger a custom Chef recipe that shuts down services.

Illustration of PHP App deployment using AWS OpsWorks



Part 2. Cloud Platform Enablement

- OpenStack
- Public Cloud Example
 - Amazon Cloud Enablement Services

Enablement - OpenStack

- An overview of open source cloud technologies
- Introduction to OpenStack and its components
- OpenStack networking options – which one is right for your environment?
- Deploying and Operating OpenStack

Cloud open source is crowded



Drupal™

SaaS

AppScale



PaaS



OPENSIFT

cloudstack
open source cloud computing

EUCALYPTUS

OpenNebula.org
The Open Source Solution for Data Center Virtualization

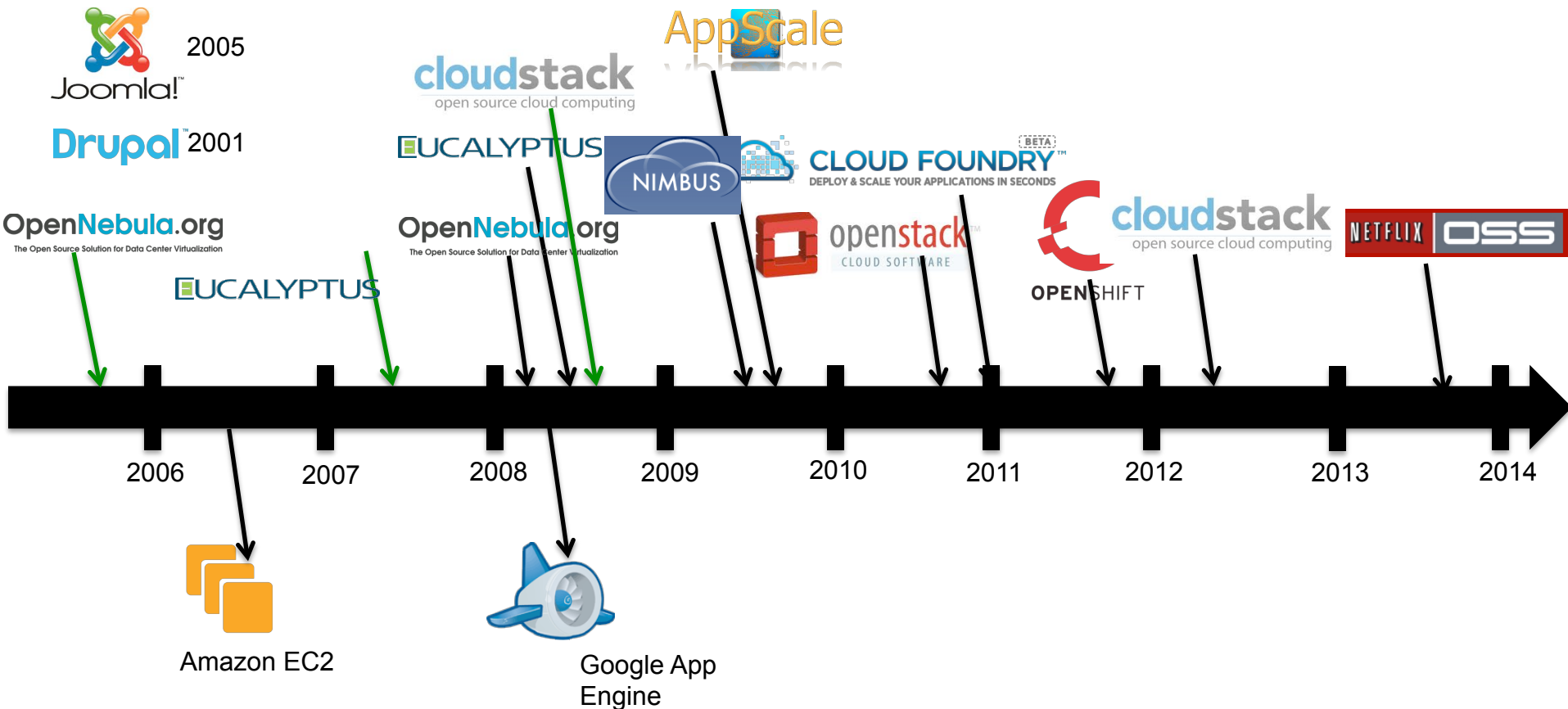


openstack™
CLOUD SOFTWARE



IaaS

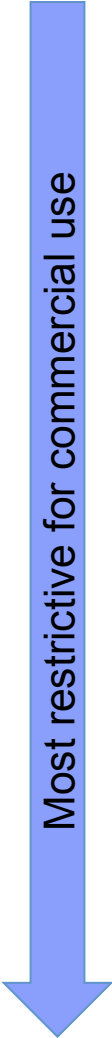
Time line for cloud open source



IaaS and PaaS open source projects trail their commercial counterparts by ~ 2 - 3 years

An open source license primer

- BSD / MIT
 - <http://www.opensource.org/licenses/bsd-license.php>
 - Least restrictive
- Apache (v2.0, v1.1)
 - <http://www.opensource.org/licenses/apache2.0.php>
 - <http://www.opensource.org/licenses/apachepl-1.1.php>
 - Can include code in a commercial product
- LGPL
 - <http://opensource.org/licenses/lgpl-3.0.html>
 - <http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html>
 - Allows dynamic linking of non-GPL / non-LGPL code to LGPL code; otherwise, almost the same as GPL.
- GPL (v2, v3)
 - <http://opensource.org/licenses/gpl-3.0.html>
 - <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>
 - Source code and binaries incorporating GPL code and binaries must be released under GPL.



Most restrictive for commercial use

Licenses for cloud open source



SaaS

GPL



BSD



Apache v2.0



OPENSIFT

Apache v2.0

PaaS



Apache v2.0



GPL



Apache v2.0



Apache v2.0



Apache v2.0

IaaS

Apache v2.0 is the most used license

Languages written in



SaaS



Python
Ruby
Go



Ruby



OPENSHIFT
Ruby, PHP etc

PaaS



Java
Python
Shell scripts



Java
C/C++
Python
Perl
Shell scripts



C / C++
Ruby
Shell scripts
Java



Python
Shell scripts



Java
Python

IaaS

What is OpenStack?

- OpenStack is an open source software for deploying infrastructure-as-a-service clouds
 - Under active development by a community of developers
 - Create or delete VMs, attach or destroy block storage, VM migration, VM snapshots, create or delete virtual networks etc
- First release October 2010
- License: Apache 2.0
- Language: Python
- Governance: OpenStack foundation

OpenStack foundation

- <http://www.openstack.org/foundation/>
- Technical committee
 - Responsible for technical stewardship of OpenStack
 - 13 total members (5 direct elects, 8 project technical leads)
- Board of directors
 - Provides strategic and financial oversight of foundation
 - Platinum, gold, individual
 - 8 platinum, 8 gold, 8 individual
- User committee
 - User advocacy and feedback

Where is OpenStack being used?

Offerings (not a complete list)

- Rackspace public and private cloud
- HP public cloud
- DreamHost public cloud
- IBM Smart Cloud Orchestrator (private cloud)
- Red Hat (private cloud)

Internal clouds (not a complete list)

- Bloomberg internal cloud
- Paypal internal cloud
- Mercado Libre

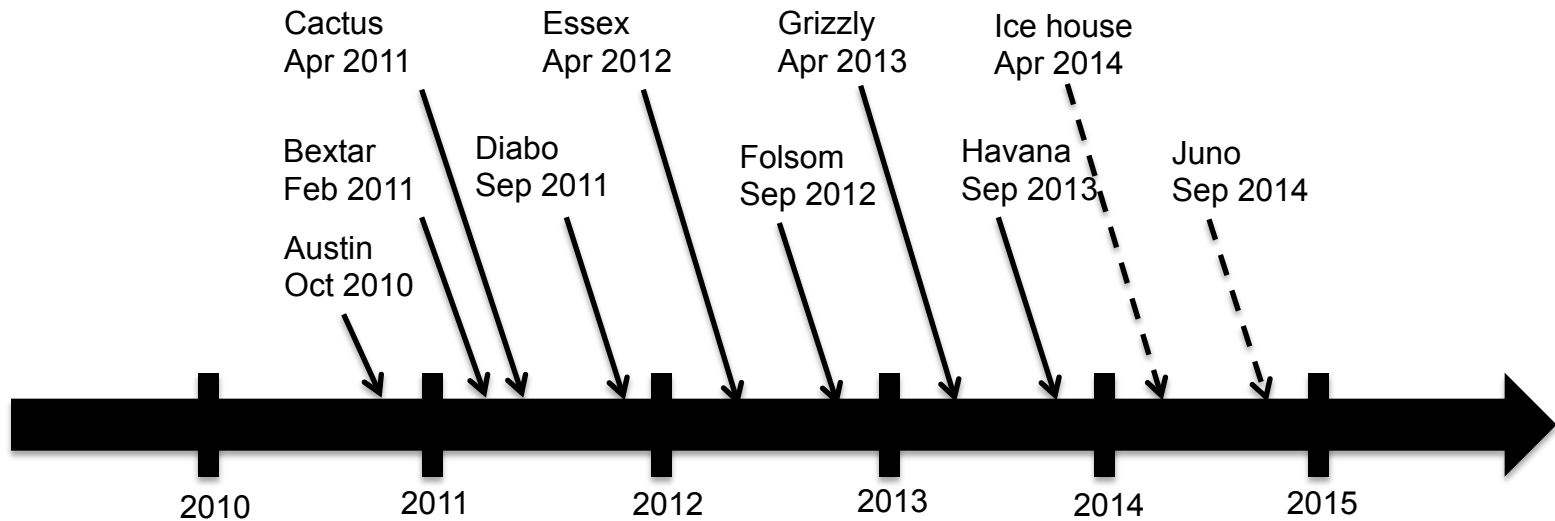
Education (not a complete list)

- CERN
- NICTA

OpenStack != IaaS Cloud

OpenStack provides the software (60-70%) of the work. The remaining work is configuring and running it to meet your needs.

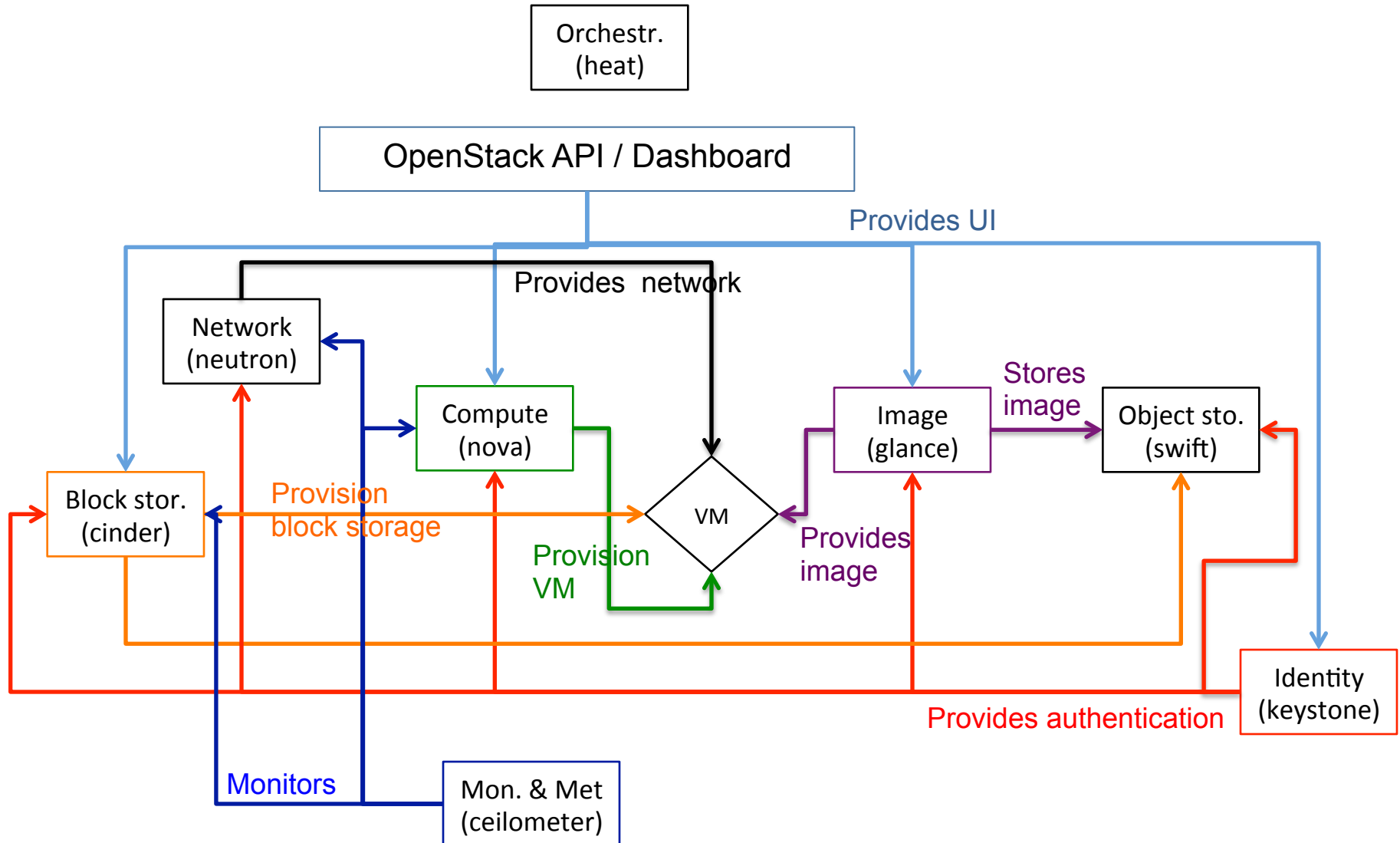
OpenStack releases



OpenStack conceptual architecture

- Compute (nova)
 - Start and manage virtual instances Analogous to Amazon EC2, Rackspace Cloud Servers for compute; S3 and CloudFiles for storage.
- Block storage (cinder)
 - Manages block storage
- Image service (glance)
 - Storage, lookup and retrieval system for VM images
- Identity management (keystone)
 - A unified identity management across nova, swift, and glance
- Network (neutron)
 - virtualizing network
- Dashboard (horizon)
 - A web portal
- Object storage (swift)
 - Store objects in a large capacity system
 - Analogous to Amazon S3 or Rackspace cloud files
- Orchestration (heat)
 - Creates a bunch of VMs based on templates specified by a user
- Monitoring and Metering (ceilometer)
 - Provides the monitoring and metering service

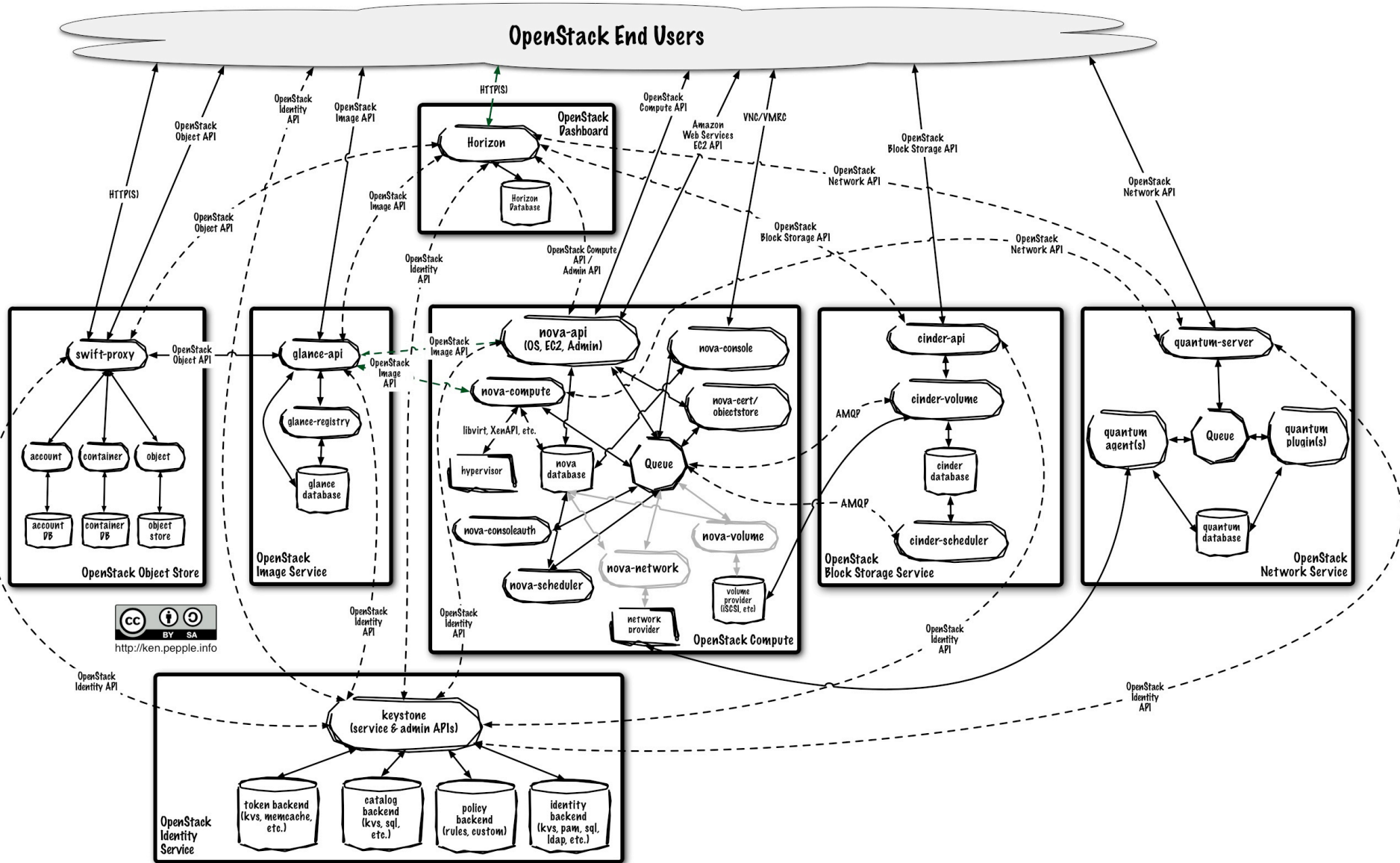
OpenStack conceptual architecture



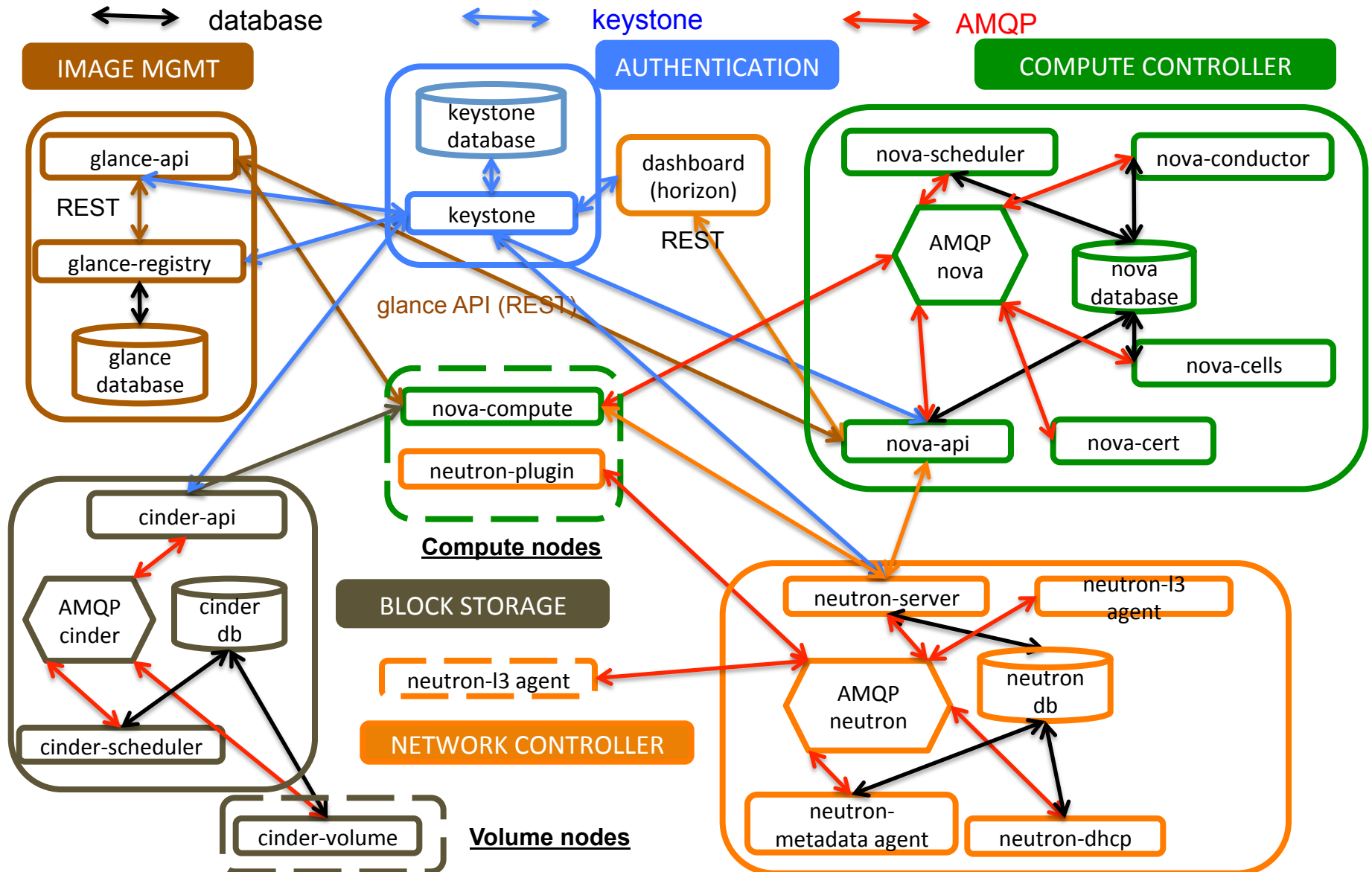
OpenStack component maturity (relative)

- Compute (nova) – reasonably mature
 - Start and manage virtual instances Analogous to Amazon EC2, Rackspace Cloud Servers for compute; S3 and CloudFiles for storage.
- Block storage (cinder) – reasonably mature
 - Manages block storage
- Image service (glance) – reasonably mature
 - Storage, lookup and retrieval system for VM images
- Identity management (keystone) – mature
 - A unified identity management across nova, swift, and glance
- Network (neutron) – evolving
 - virtualizing network
- Dashboard (horizon) – evolving
 - A simple web portal
- Object storage (swift) – reasonably mature
 - Store objects in a large capacity system
 - Analogous to Amazon S3 or Rackspace cloud files
- Orchestration (heat) – evolving
 - Creates a bunch of VMs based on templates specified by a user
- Monitoring and Metering (ceilometer) – not mature
 - Provides the metering service

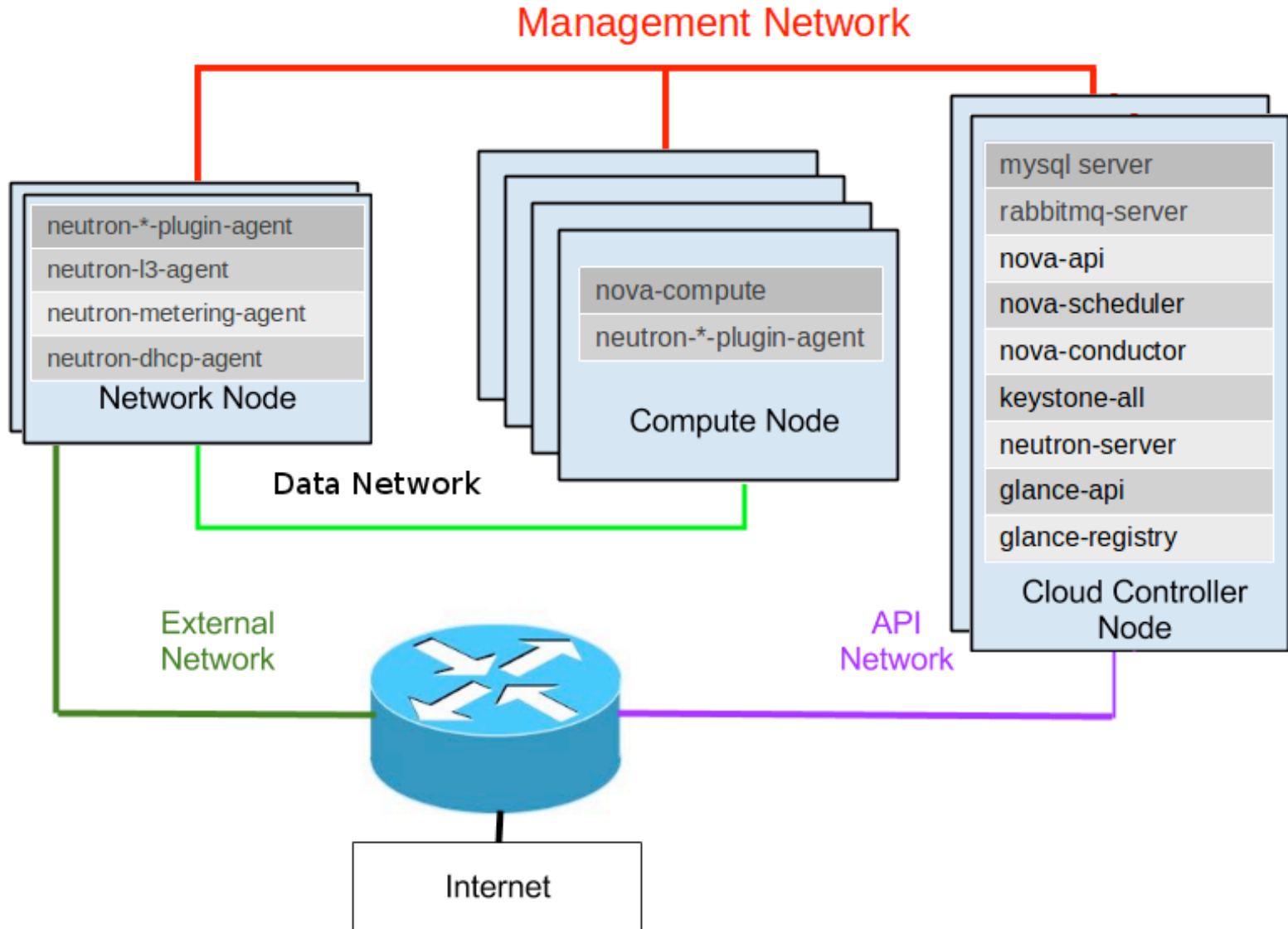
OpenStack logical architecture



OpenStack logical architecture (havana+neutron+cinder)



OpenStack deployment architecture



OpenStack demo

- Projects
- Flavors
- VMs

Brewing complexity: Evolution of OpenStack loc *

	Released	Nova	Cinder	Glance	Keystone	Quantum / Neutron	Swift	Total
Austin	Oct 2010	17,288					12,979	30,627
Bexar	Feb 2011	27,734		3,629			16,014	47,377
Cactus	Apr 2011	43,947		4,927			16,665	65,539
Diablo	Sep 2011	66,395		9,961	12,451		15,591	91,947
Essex	Apr 2012	87,750		15,698	11,555		17,646	149,596
Folsom	Sep 2012	103,637	31,241	20,271	13,939	42,118	19,114	230,320
Grizzly	Apr 2013	120,968	49,797	21,261	20,071	60,485	23,035	321,081
Havana	Sep 2013	158,999	88,558	35,847	41,096	118,699	31,496	474,695

Methodology

```
wc -l `find . | grep -E '*\.py' | grep -v test | grep -v 'doc'`
```

```
wc -l `find . | grep -E '*\.sh' | grep test | grep -v 'doc'`
```

> 50 configuration files, > 2000 parameters

* CRLF and not python loc

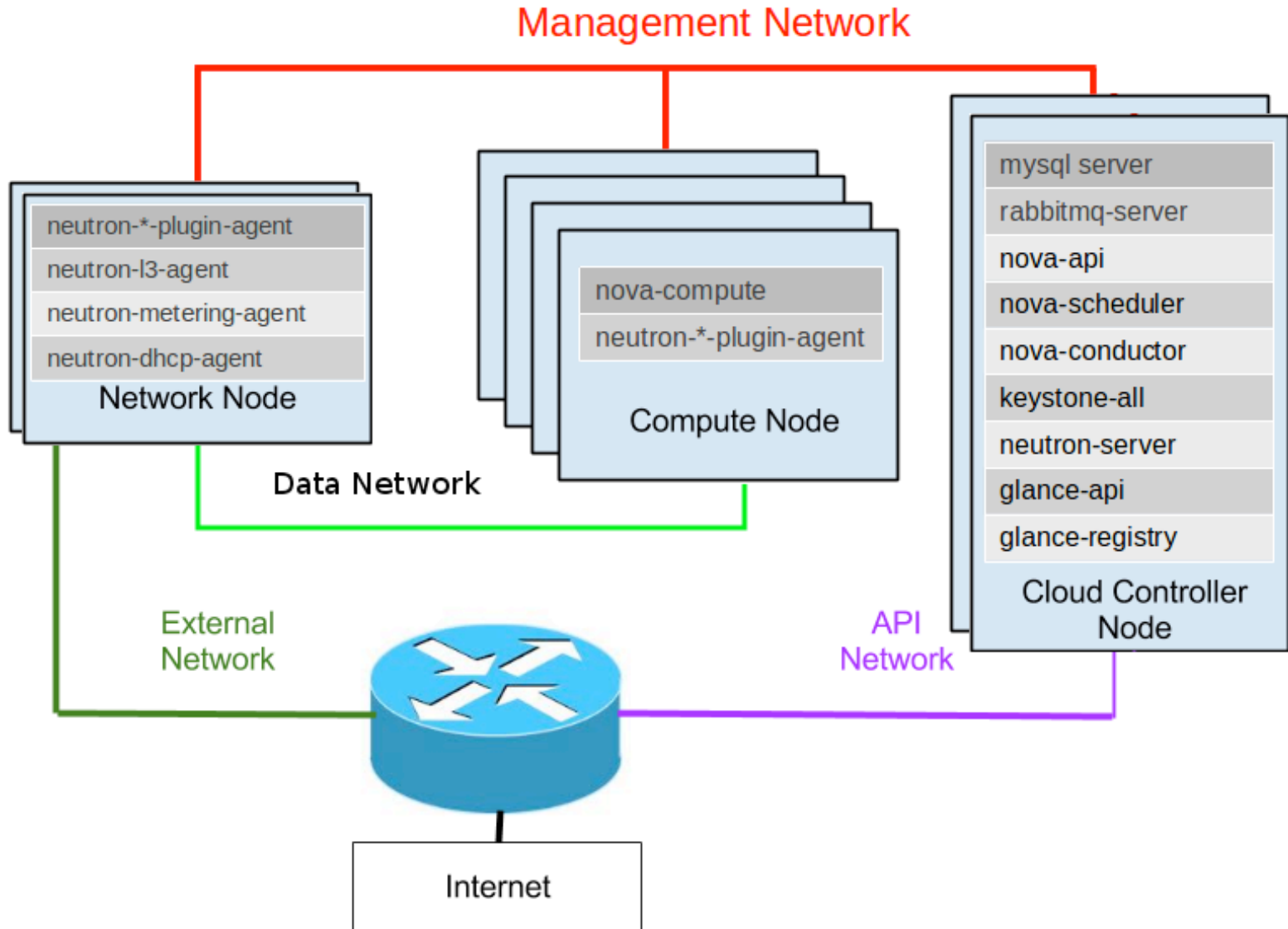
Enablement - OpenStack

- An overview of open source cloud technologies
- Introduction to OpenStack and its components
- OpenStack networking options – which one is right for your environment?
- Deploying and Operating OpenStack

OpenStack networking

- Neutron
 - VLAN – use VLANs to segment different projects
 - GRE – Provide isolation by creating full mesh tunnels between end points
 - Flat – Create a network that directly maps onto physical network
 - Custom plugin (e.g., Nicira, IBM SDN-VE, Cisco Nexus)
- Nova-network
 - VLAN – use VLANs to segment different projects
 - Flat – Create a network that directly maps onto physical network
- Limitations
 - For Provider or per-tenant router types, OpenStack mixes NAT with layer 3 forwarding.

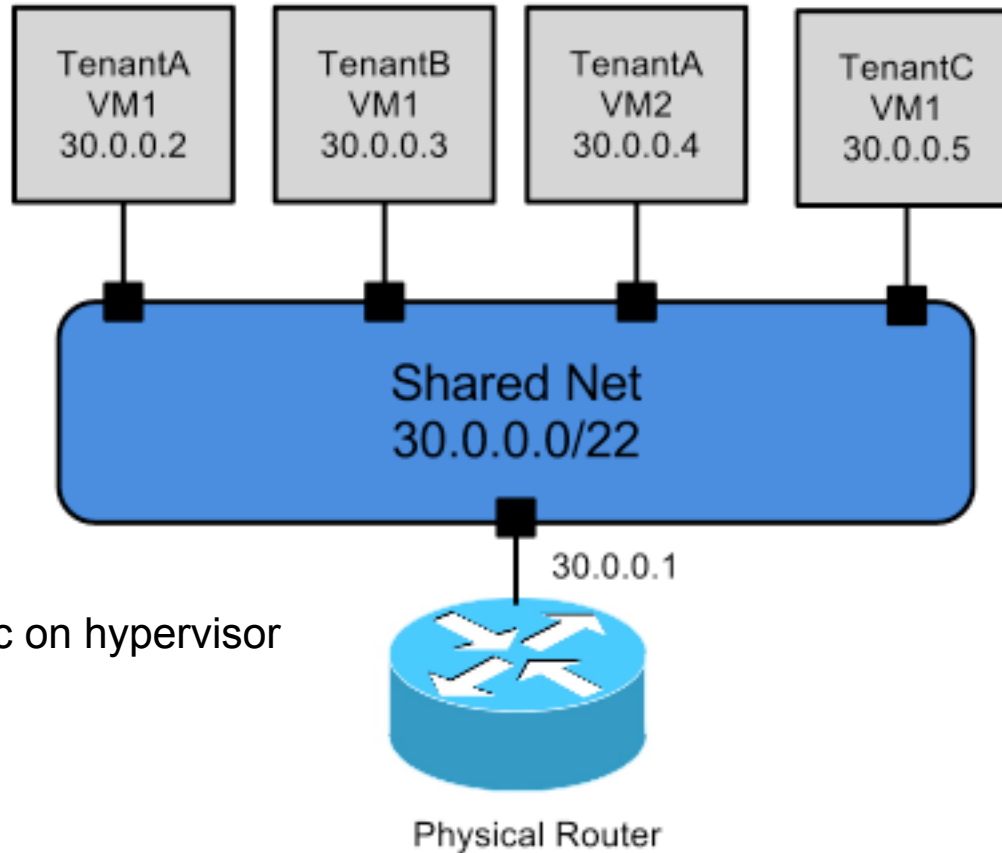
OpenStack networking



Neutron deployment use cases

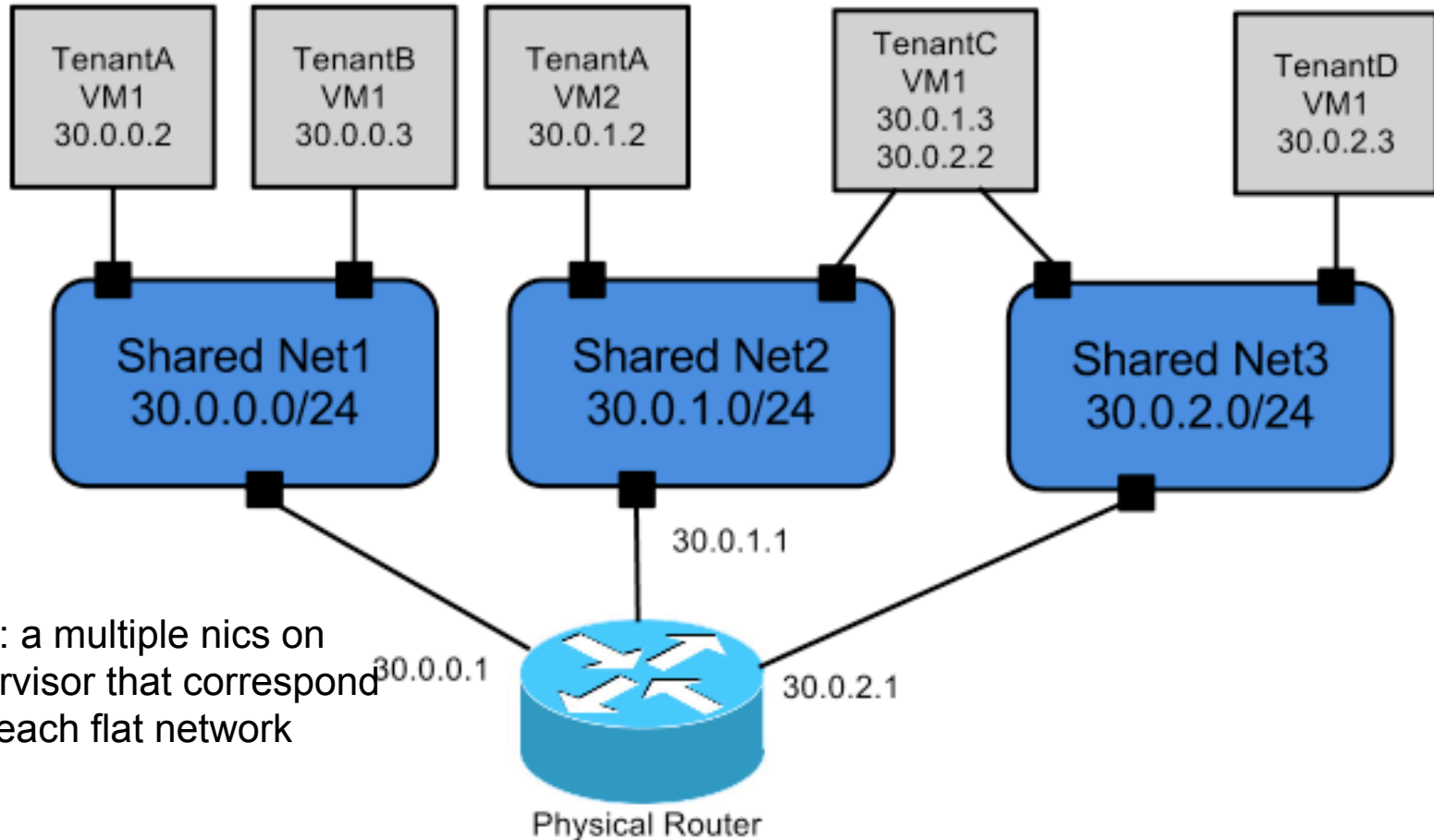
- Single flat network
- Multiple flat network
- Mixed flat and private network
- Provider router with private networks
- Per-tenant router with private networks

Single flat network



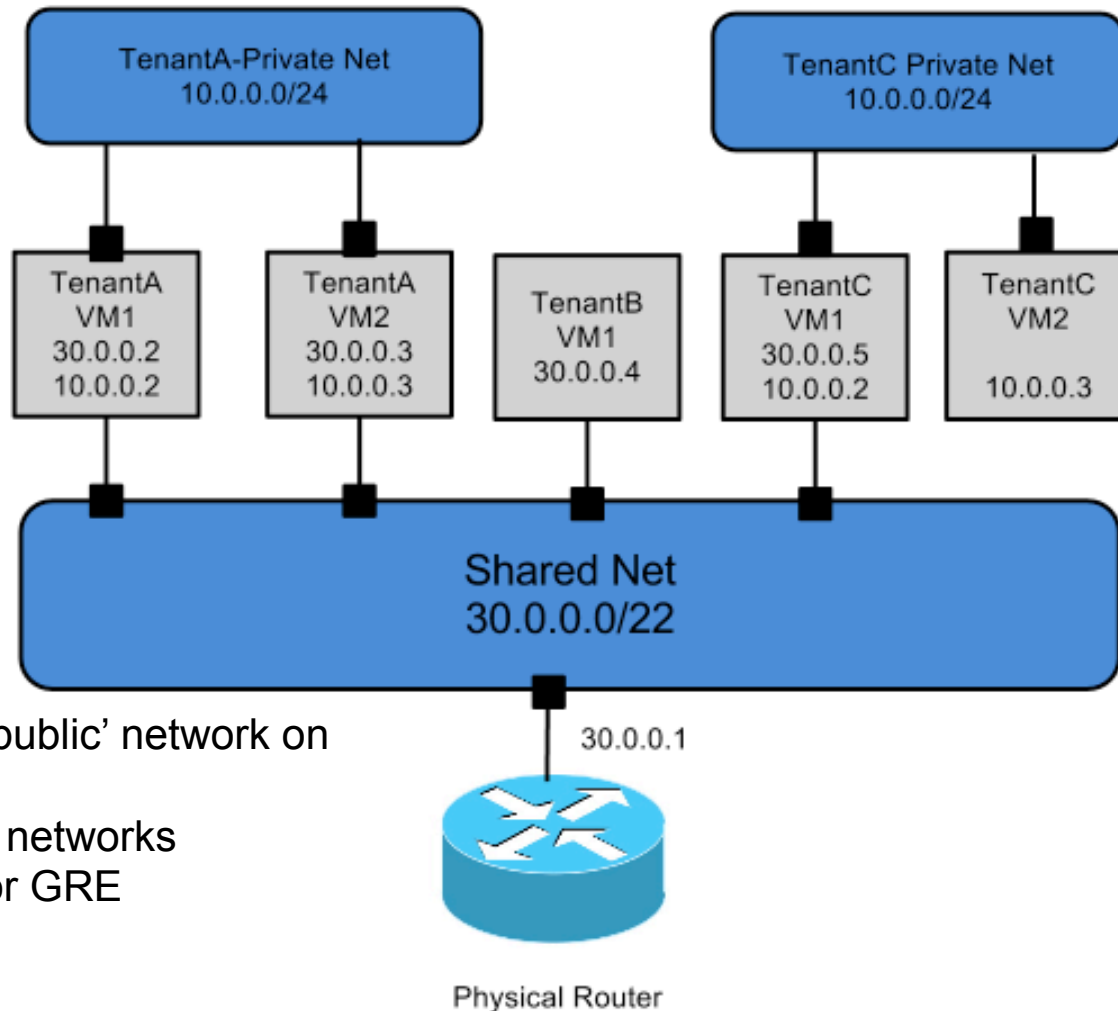
REQ: a single nic on hypervisor
for VMs

Multiple flat network



REQ: a multiple nics on hypervisor that correspond with each flat network

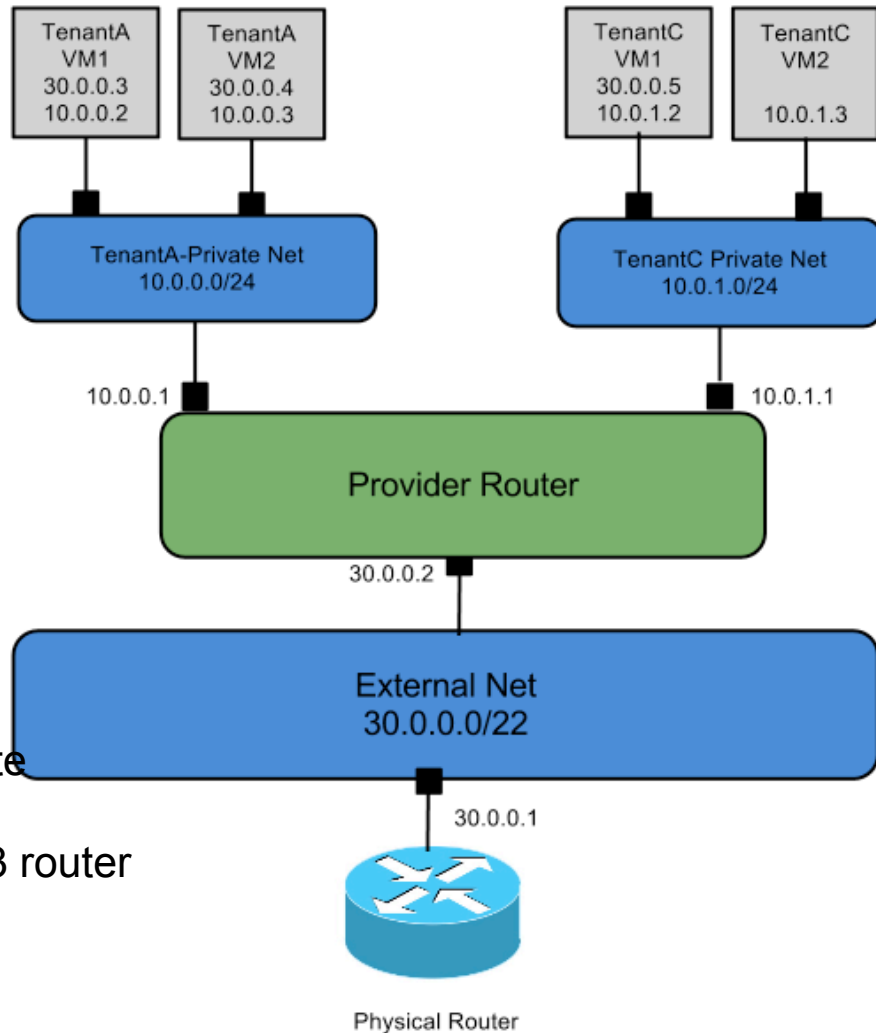
Mixed flat and private network



REQ: a nic for 'public' network on hypervisors.

A nic for private networks through VLAN or GRE

Provider router with private networks



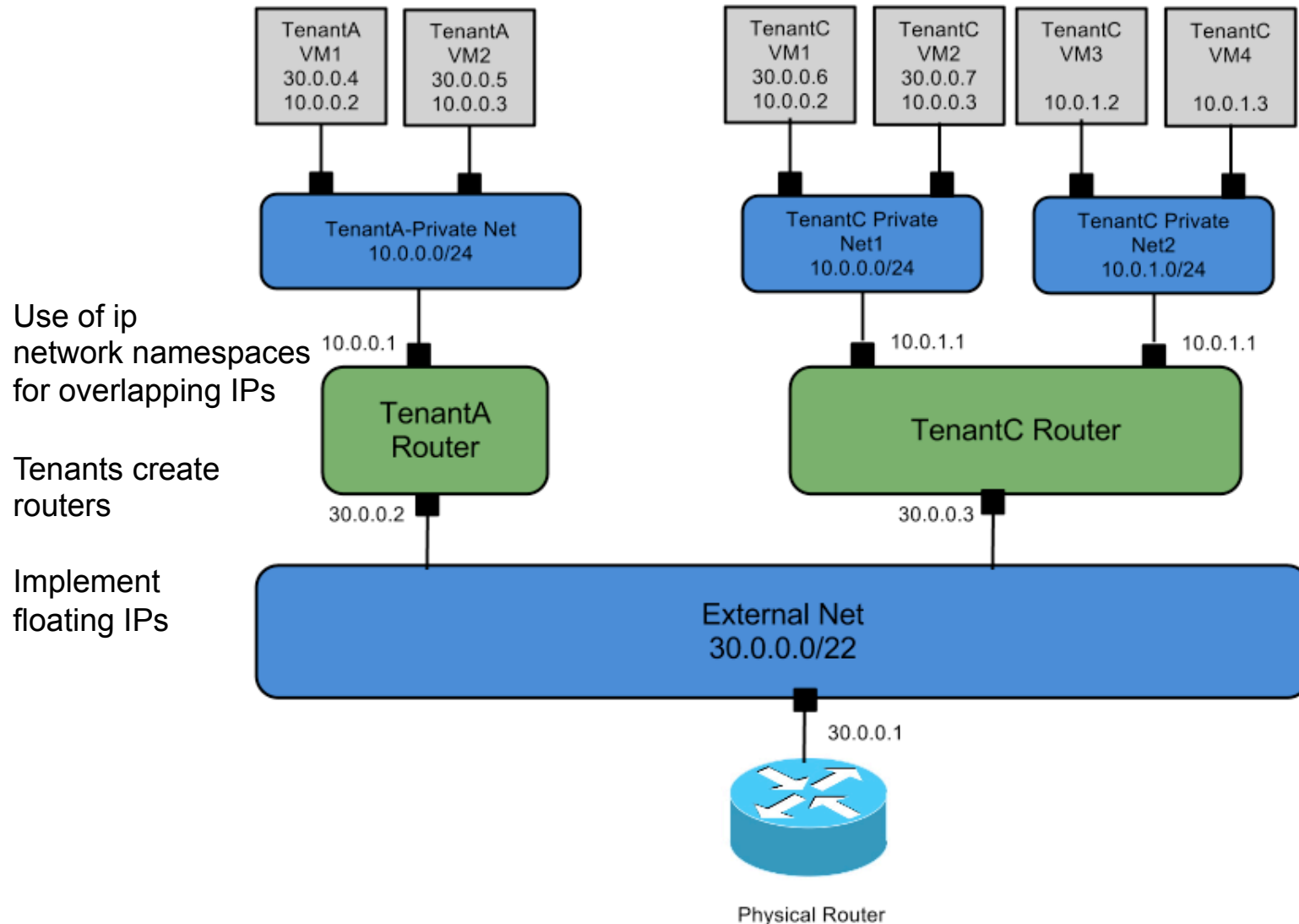
Provider router created by OpenStack cloud administrator

Implement elastic / floating IPs

REQ: a nic for private network.

REQ: two nics on L3 router

Per-tenant routers with private networks



Enablement - OpenStack

- An overview of open source cloud technologies
- Introduction to OpenStack and its components
- OpenStack networking options – which one is right for your environment?
- Deploying and Operating OpenStack

Deploying and Operating OpenStack

- OpenStack != cloud
- Know thy requirements
- Multiple networking options in OpenStack can accomplish similar objectives
- Developer or operator (big difference)
 - Devstack (developer)
 - Operator (GUI based life cycle management tooling)

How do I get started?

- Developer
 - Ubuntu (Precise), use devstack
- Small test deployment
 - Get baremetal or virtual machines
 - If you do not have any bare metal machines, you can obtain them through SoftLayer
- Operator
 - Understand business requirements
 - Translate them into simplest configurations for OpenStack

OpenStack deployment tools

Tool	Description	Status
Deploy by hand	Not recommended	Do it your self
Puppet tooling	Puppet to install OpenStack	Open source, Missing GUI. Requires a baremetal install
Chef tooling (stack forge)	Chef to install OpenStack	Open source, Missing GUI. Requires a baremetal install
Red Hat Pack stack	Automated installation	Open source, not very flexible
Mirantis Fuel	GUI based automated installation	Open source
Foreman	Machine life cycle management	Open source
Dell's crow bar	Bare metal and OpenStack installation	Open source

OpenStack deployment planning questionnaire (1/4)

Questions (size and storage)									
1	What is the size of your deployment?								
2	< 1U rack	REC: A single availability zone or host aggregate							
3	> 1U rack	REC: Multiple availability zones							
4	Do you want to enable live VM migration?								
5	Y	Will VMs have ephemeral disks?							
6		Y	Do you have external storage available for VM ephemeral disks?						
7			Y	REC: Use NFS, Gluster FS, GPFS with external storage					
8			N	Will you hypervisor local storage					
9				Y	Do you want hypervisor evacuation for maintenance?				
10					Y	REC: Use GPFS-FPO with replication factor of 2 or Ceph NOT REC: Use KVM block migration. SLOW			
11					N	REC: Use NFS NOT REC: Use KVM block migration. SLOW			
12	N	REC: You must use block storage (or boot from volume)							
13	Can VMs attach to block storage device?								
14	Y	What type of block storage device do you have? iSCSI, Fiber							
15	N	If you answered no to (5), you must choose local or block storage for VMs If you answered yes to (5), your VMs will only have local storage							

OpenStack deployment planning questionnaire (2/4)

Questions (storage and oversubscription)		
16	If you selected (Y) to “Will you use hypervisor local storage for VM ephemeral disks? (8)” How many VMs per hypervisor? (for mixed load)	
	2-4	REC: Use 1 spindle (rule of thumb. 4 VMs per disk spindle, 1/2-1 for Hadoop)
	4-8	REC: Use 1-2 spindles
	8-16	REC: Use 2-4 spindles
	16-32	REC: Use 4-8 spindles
	32-64	REC: Use 8-16 spindles
17	Will you oversubscribe memory?	
	Y	REC: Use live migration or a mechanism for terminating machines during overload
	N	Do nothing
18	Will you oversubscribe disk?	
	Y	REC: use live migration or a mechanism for terminating machines during overload

OpenStack deployment planning questionnaire (3/4)

	Questions (users and projects)
19	Will your OpenStack deployment have multiple projects?
20	Will you use your LDAP directory for linking with OpenStack authentication?

OpenStack deployment planning questionnaire (4/4)

Questions (networking)		
21	Will there be private subnet (shared or per tenant)?	
22	If (Y) to 21, will a user create private subnets that may overlap with IP subnet range of another user?	
23	Y	REC: Use per tenant router with L3 agent or preferably a network plugin with namespaces
24	N	Will users have floating IP addresses? REC: Do not use ip network namespaces
25	Y	REC: use L3 agent or a network plugin
26	N	REC: use Flat, and VLAN networking (either nova-network or neutron) / GRE (neutron)
27	If (N) to 21, then use single Flat public network. No VLAN or GRE dependency	
28	Are VLANs available?	
	Y	REC: Use VLANs for VM private network per project
	N	REC: If < 1 Rack of 1U machines, use GRE

OpenStack common problems

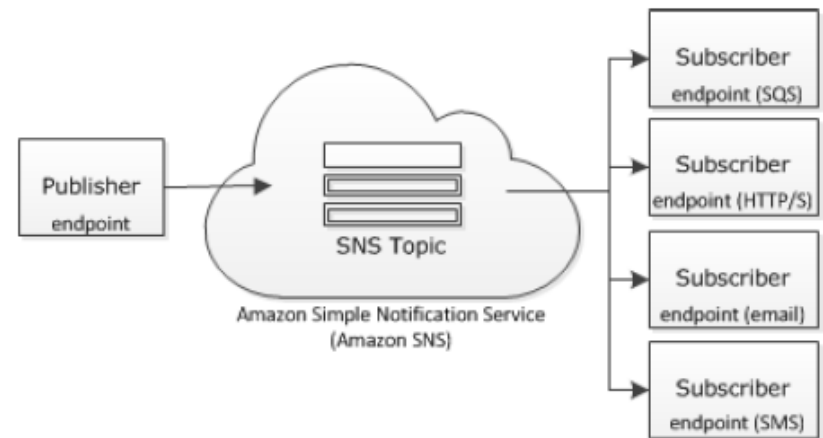
- VMs not getting IP addresses
- Orphan VMs in hypervisors
- Image cache not being cleared on hypervisors
- VNC console not working
- Performance of VMs – function of deployment architecture

AWS Application Enablement Services

- Simple Notification Service
- Elastic Beanstalk
- CloudFront
- Cloud Formation
- Elastic Map reduce

Amazon Simple Notification Service (SNS)

- Amazon Simple Notification Service (Amazon SNS) is a fast, flexible, fully managed push messaging service.
- Amazon SNS makes it simple and cost-effective to push to mobile devices such as iPhone, iPad, Android, Kindle Fire, and internet connected smart devices, as well as pushing to other distributed services and Amazon Simple Queuing Service (SQS) or any HTTP end-point.
- Typical use cases
 - Application and system alerts
 - Push email and text messaging
 - Mobile push notification
 - Complex distributed services and application via SQS



Create and Subscribe Topics Flow

- Sign up for aws.amazon.com/sns
- Create a topic

Create New Topic [Cancel]

A topic named **MyTopic** has been created.

Topic Details

All Topic Actions [Publish to Topic]

Topic ARN: arn:aws:sns: : :MyTopic

Topic Owner: AWS Account ID

Region:

Display Name: DisplayNameHere

[Create New Subscription] [Delete Subscriptions] [View/Edit Delivery Policy]

```
//create a new SNS client and set endpoint
AmazonSNSClient snsClient = new AmazonSNSClient(new ClasspathPropertiesFileCre
dentialsProvider());
snsClient.setEndpoint("http://sns.us-east-1.amazonaws.com");

//create a new SNS topic
CreateTopicRequest createTopicRequest = new CreateTopicRequest("MyNewTopic");
CreateTopicResult createTopicResult = snsClient.createTopic(createTopicRequest);
//print TopicArn
System.out.println(createTopicResult);
//get request id for CreateTopicRequest from SNS metadata
System.out.println("CreateTopicRequest - " + snsClient.getCachedResponse
Metadata(createTopicRequest));
```

- Subscribe to a topic
 - Need the ARN for a topic to subscribe
 - Provide an end-point to receive notification

Subscribe [Cancel]

Topic ARN []

Protocol [HTTPS]

Endpoint []

[Cancel] [Subscribe]

```
//subscribe to an SNS topic
SubscribeRequest subRequest = new SubscribeRequest(topicArn, "email",
"name@example.com");
snsClient.subscribe(subRequest);
//get request id for SubscribeRequest from SNS metadata
System.out.println("SubscribeRequest - " + snsClient.getCachedResponse
Metadata(subRequest));
System.out.println("Check your email and confirm subscription.");
```

Amazon Elastic Beanstalk: Application Container

- Elastic Beanstalk is an application deployment, configuration and management platform on AWS
- Deployment, capacity provisioning, load balancing, auto-scaling are hidden away from application creator view
- Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server for Node.js, Apache Tomcat for Java, PHP and Python, Passenger for Ruby, IIS 7.5 for .NET
- Create and deploy an application using Elastic Beanstalk
 - <https://console.aws.amazon.com/elasticbeanstalk>

CloudFront: CDN Service from AWS

- <http://docs.amazonwebservices.com/AmazonCloudFront/latest/DeveloperGuide/Overview.html>
- CloudFront is a web service for content delivery. It makes it easier for you to distribute content to end users quickly, with low latency and high data transfer speeds.
- CloudFront delivers your content through a worldwide network of edge locations. End users are routed to the nearest *edge location*, so content is delivered with the best possible performance. CloudFront works seamlessly with the Amazon Simple Storage Service, which durably stores the original, definitive versions of your files.

Cloud Front Components

- **Objects**

- Objects are the files you want CloudFront to deliver. This typically includes web pages, images, and digital media files, but can be anything that can be served over HTTP or a version of RTMP.

- **Origin Server**

- An origin server is the location where you store the original, definitive version of your objects. For CloudFront, your origin server is either an Amazon S3 bucket that you own, or a custom origin. For more information about origin servers, see [The Origin Server](#).

- **Distributions**

- After you store your objects in your origin server, how do you get CloudFront to recognize them? You create a *distribution*, which is a link between your *origin server* and a domain name that CloudFront automatically assigns. If your origin is an Amazon S3 bucket, you use this new domain name in place of standard Amazon S3 references, for example, `http://myawsbucket.s3.amazonaws.com/image.jpg` would instead be `http://somedomainname.cloudfront.net/image.jpg`.

- Distributions can incorporate CNAME aliases you want to use. For more information, see [Using CNAMEs](#).

- A *streaming distribution* is like a distribution except that a streaming distribution delivers digital media over a Real-Time Messaging Protocol (RTMP) connection (instead of HTTP). Adobe's Flash Media Server can serve RTMP and the streaming distribution can be used as it downloads. The objects stored in Amazon S3 are different according to the distribution type.

- **Edge Locations**

- An *edge location* is a geographical site where CloudFront caches copies of your objects. When an end user requests one of your objects, CloudFront decides which edge location is best able to serve the request. If the edge location doesn't have a copy, CloudFront goes to the origin server and puts a copy of the object in the edge location.
- For the locations of CloudFront edge locations, go to the [CloudFront Details](#) page.

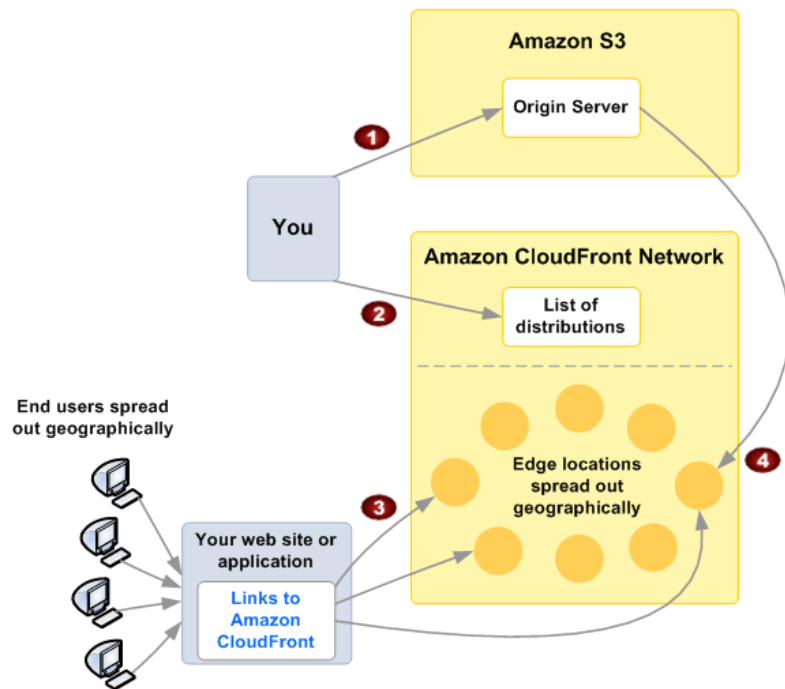
- **Expiration**

- By default, an object expires after being in an edge location for 24 hours. After the object expires, CloudFront no longer serves that particular object and must get a new copy of that object from the origin server to serve to end users. The minimum expiration time is 0 seconds; there isn't a maximum expiration time limit. For more information about expiration, see [Specifying When an Object Expires](#).

- **Eventual Consistency**

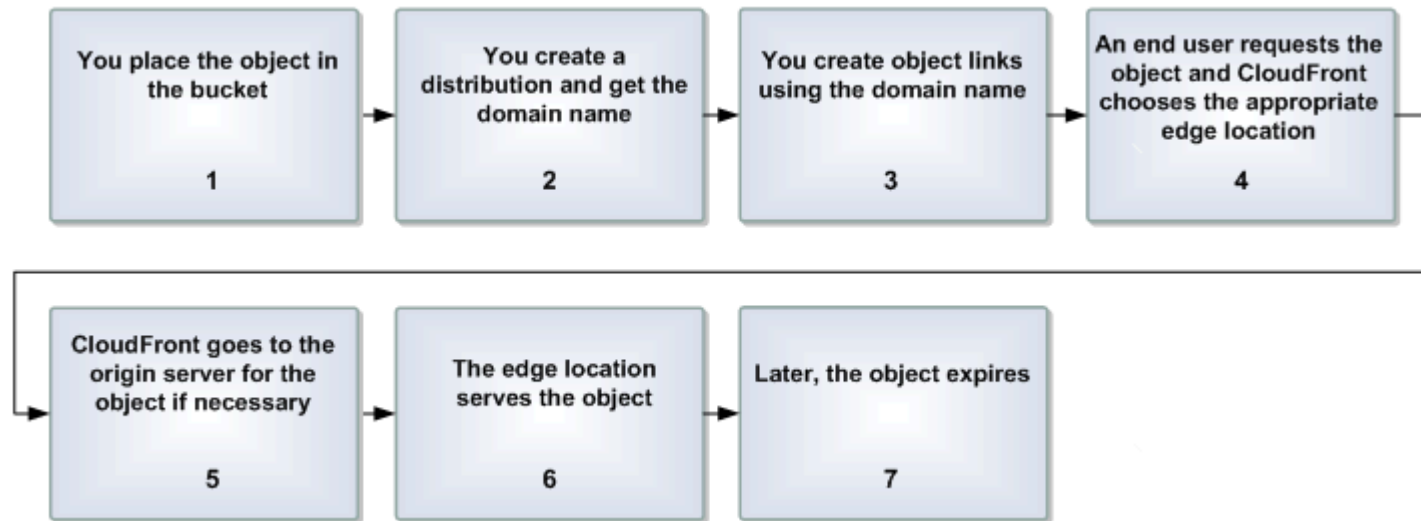
- When you successfully create, modify, or delete a distribution using the CloudFront *control API*, it takes time for your changes to propagate throughout the CloudFront system. This information about the distribution is eventually consistent, but an immediate request to the control API to get that distribution's information might not show the change. Consistency is usually reached within minutes, but a high system load or network partition might increase this time. The control API lets you determine when your changes have been fully deployed.

Cloud Front: Architecture



	Communication Between	Reason
1	You and Your Origin	To put objects in the origin server, manage them, and delete them. If using Amazon S3, this is the normal interaction you have with Amazon S3 to manage buckets and objects. This relationship is covered in the Amazon Simple Storage Service Developer Guide .
2	You and CloudFront	To manage your <i>distribution</i> . This communication tells CloudFront the location of your origin server and lets you configure aspects of how CloudFront should deliver your content to end users. You can communicate with CloudFront through the REST <i>control API</i> . This guide covers how to do this (for more information, see Making API Requests , and go to Amazon CloudFront API Reference).
3	End users and CloudFront	To get copies of your objects from the edge locations (via your website or application). This involves how end users link to your objects in CloudFront through your website or application. Because CloudFront (and not Amazon S3) serves the objects, you give end users links to CloudFront (instead of links to the objects in Amazon S3). For more information about the format of these links, see Format of URLs for CloudFront Objects .
4	CloudFront and Your Origin Server	To get copies of your objects from the origin server as needed. CloudFront takes care of this communication automatically when a request comes in to an edge location that doesn't already have a copy of the object.

Steps in Using Cloud Front



Process for Delivering Content

1	You place the original version of <code>image.jpg</code> in your Amazon S3 origin server bucket and make it publicly readable.
2	You create a <i>distribution</i> and get your distribution's domain name.
3	You create links to <code>image.jpg</code> in your website or web application with the CloudFront domain name.
4	When an end user requests a page that contains <code>image.jpg</code> , CloudFront determines which edge location would be best to serve <code>image.jpg</code> (in this case, we'll say it's the St. Louis location).
5	If the St. Louis edge location doesn't have a copy of <code>image.jpg</code> , CloudFront goes to the origin server and puts a copy of <code>image.jpg</code> in the St. Louis location.
6	The St. Louis edge location then serves <code>image.jpg</code> to the end user and then serves any other requests for that file at the St. Louis location.
7	Later, <code>image.jpg</code> expires, and CloudFront deletes <code>image.jpg</code> from the St. Louis location. CloudFront doesn't put a new copy of <code>image.jpg</code> in the St. Louis location until an end user requests <code>image.jpg</code> again and CloudFront determines the St. Louis location should serve the image.

Cloud Formation: Template based Application Deployment

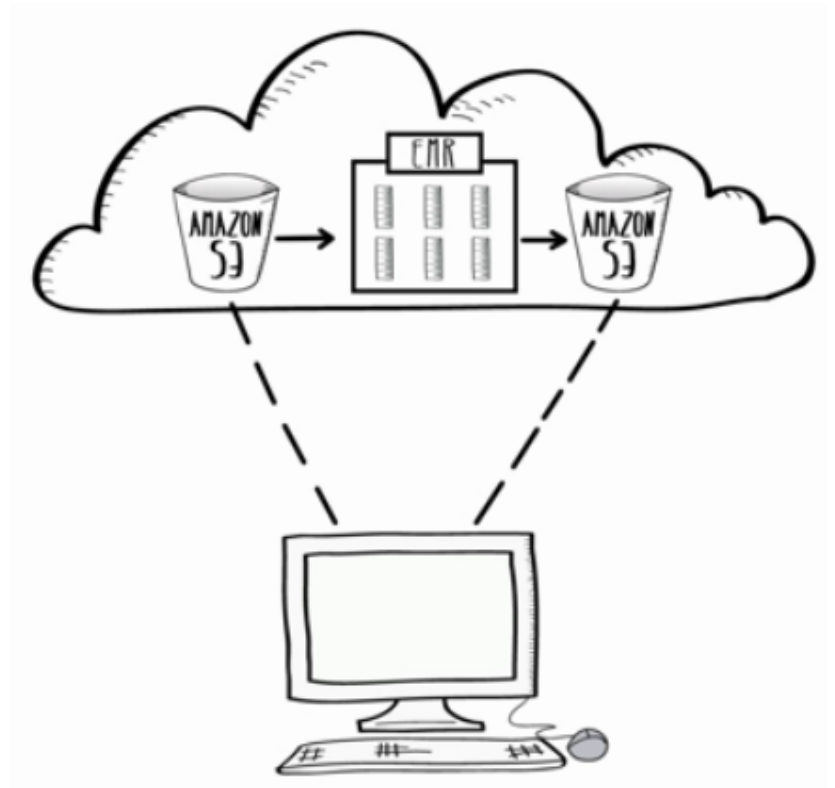
- Service that gives developers and businesses an easy way to create a collection of AWS resources and provision them in an orderly and predictable fashion.
- Simply describe the AWS resources you need to run your application in a simple text file called a template and AWS CloudFormation takes care of provisioning those resources in the right sequence and taking into account any dependencies between resources. Once provisioned, you can see all of the AWS resources you need to run your application in a single view.
- AWS CloudFormation comes with ready-to-run sample templates to deploy some common open source applications that illustrate how easy it is to get the infrastructure for an application up and running quickly.
 - These include WordPress (blog), Tracks (project tracking), Gollum (wiki) plus a wide range of sample templates to cut and paste from to create your own templates. There's a good chance that a ready-made template exists to cover what you want to do.

Cloud Formation and Application Deployment

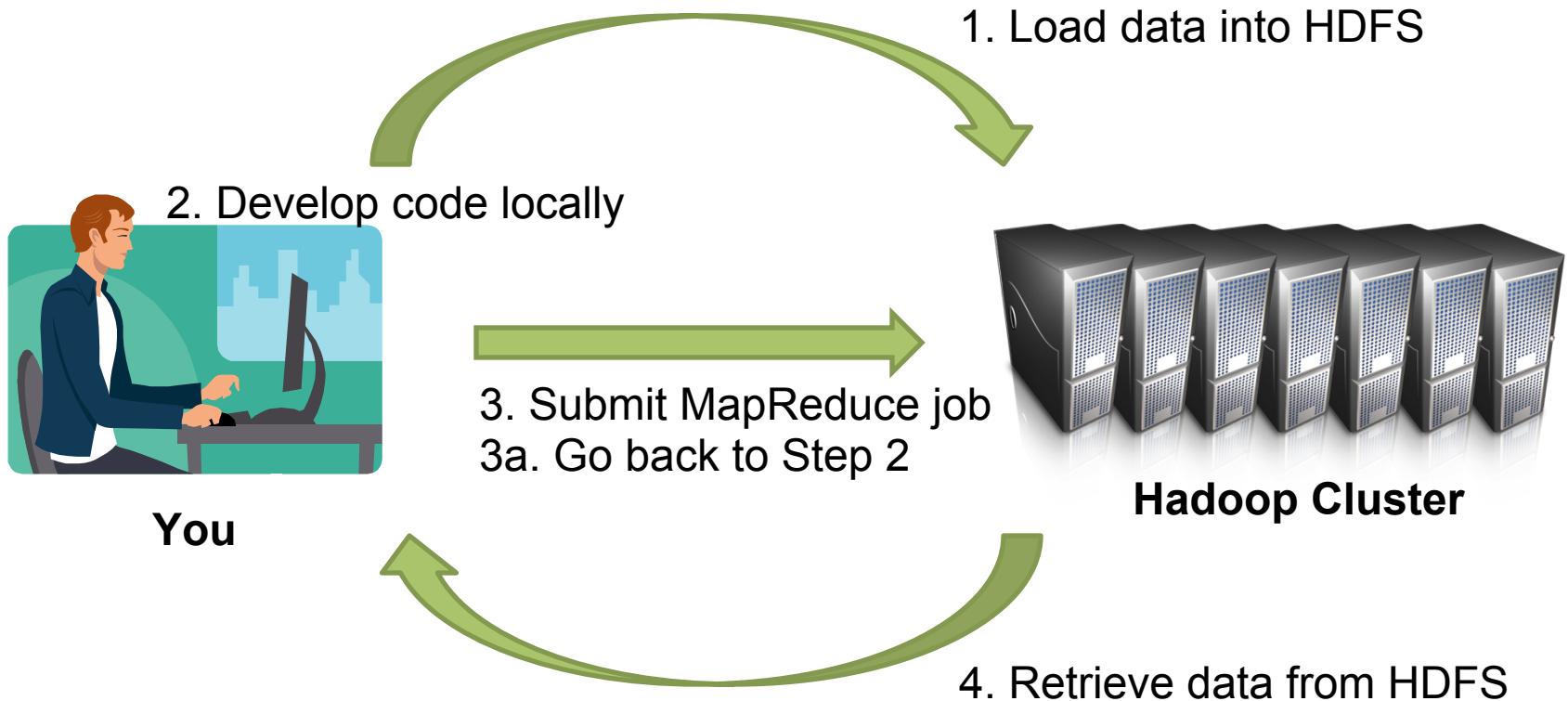
- Create a CloudFormation stack to provision the collection of resources needed by your application. You do this in two simple steps:
 - First, select the template you want to use in the CloudFormation tab of the AWS Management Console. AWS CloudFormation comes with sample templates for popular open source applications that you can deploy immediately to get started. Our sample templates illustrate how to interconnect and use multiple AWS resources in concert, following best practices for geographic redundancy, scale out, and alarming. You can also load your own custom templates that you may store in Amazon Simple Storage Service (Amazon S3) or on your local hard drive. Second, determine if you want to customize your stack by overriding template parameters (such as database or application usernames, passwords, port numbers, Amazon EC2 instance types, etc.), or simply accept the default values supplied by the template author. Once complete, AWS CloudFormation takes care of provisioning the AWS resources, creating the stack for you.
- View all the AWS resources contained in each of your stacks in the [AWS Management Console](#).
- Create additional stacks (if needed) by simply using existing templates you've created, choosing from one of our sample templates, or creating a new one from scratch. Since templates are text files that can be created and managed outside of CloudFormation, you can share them easily through email, source control repositories, or services such as Amazon S3.
- Make changes and updates to your running stacks allowing you to react to the software updates and configuration changes needed to manage your application over its lifetime. You do this in 2 simple steps: First modify your template with the configuration changes required for all of the resources in the stack. You can use your version control and change management best practices for your templates allowing you to manage changes to the configuration of your AWS infrastructure in the same way as you manage changes to the software running on it. Second, submit the updated template to AWS CloudFormation via the AWS Management Console, the command line tools or the APIs. AWS CloudFormation takes care of the rest, making the configuration changes to your stack in a controlled and predictable way.

Amazon Elastic Map Reduce (EMR)

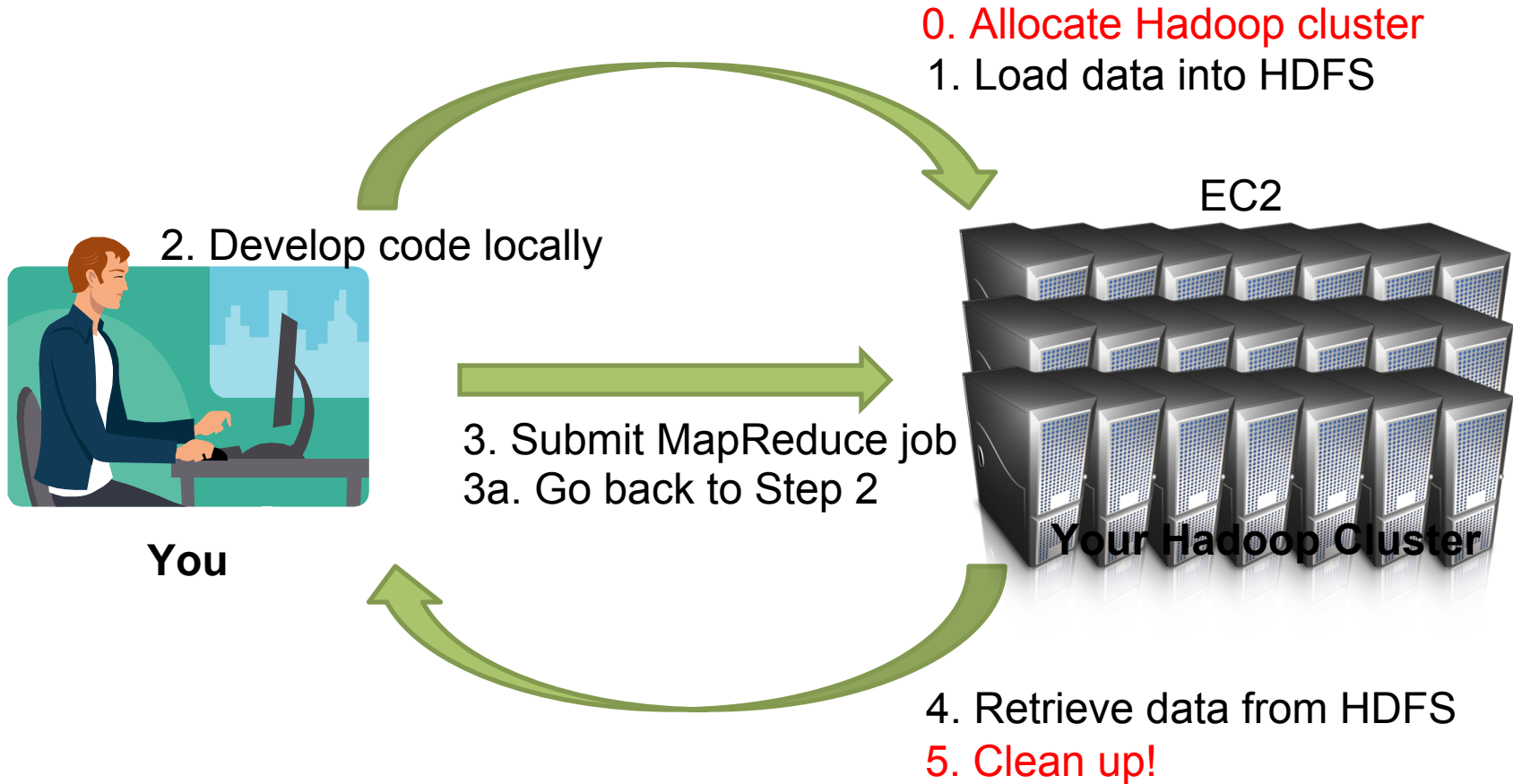
- Amazon EMR is a web service that makes it easy to quickly and cost-effectively process vast amounts of data.
- Leverages EC2 and S3 for automating the set up of Hadoop cluster for Map Reduce based computations



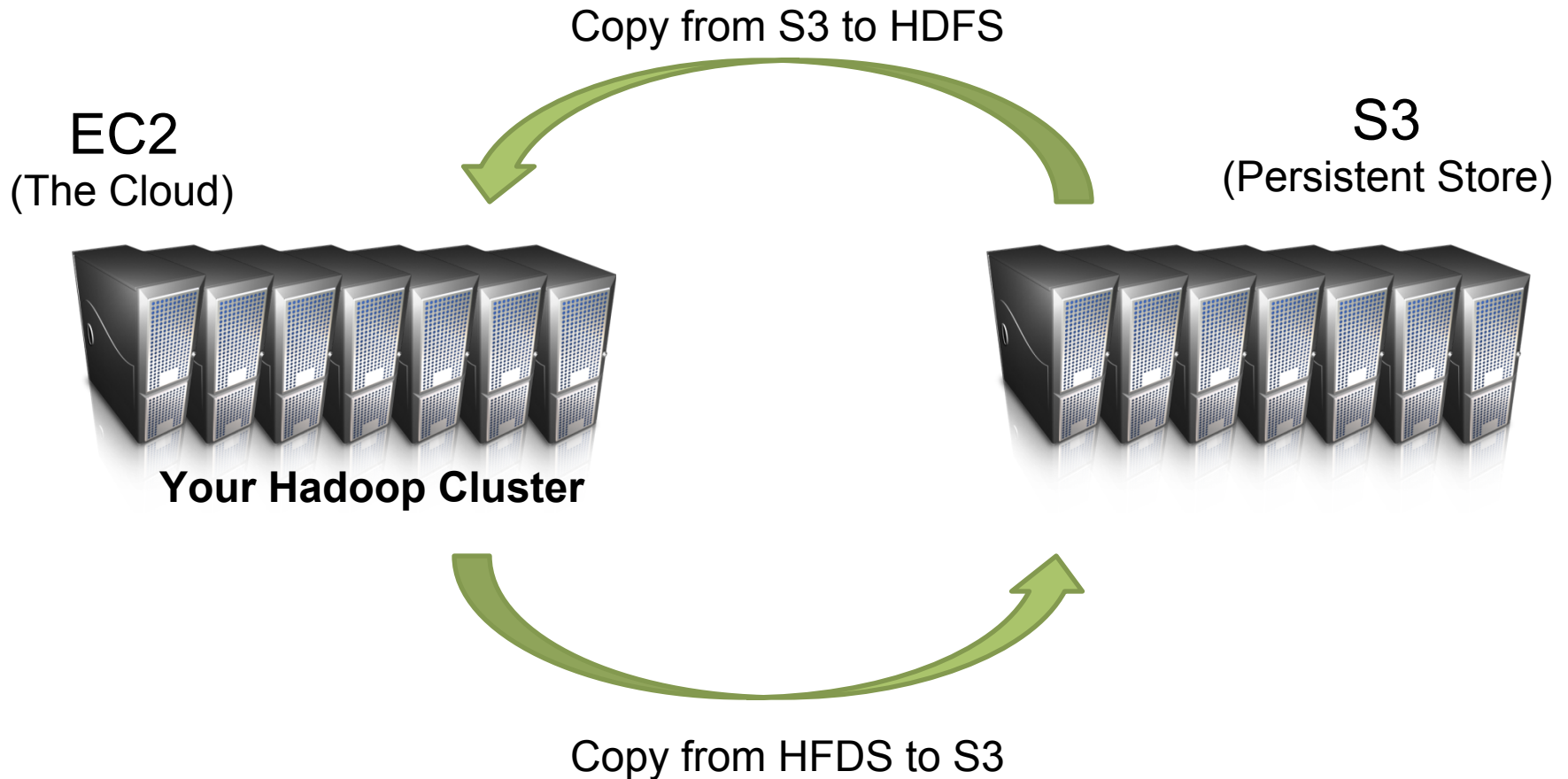
Typical Hadoop Workflow



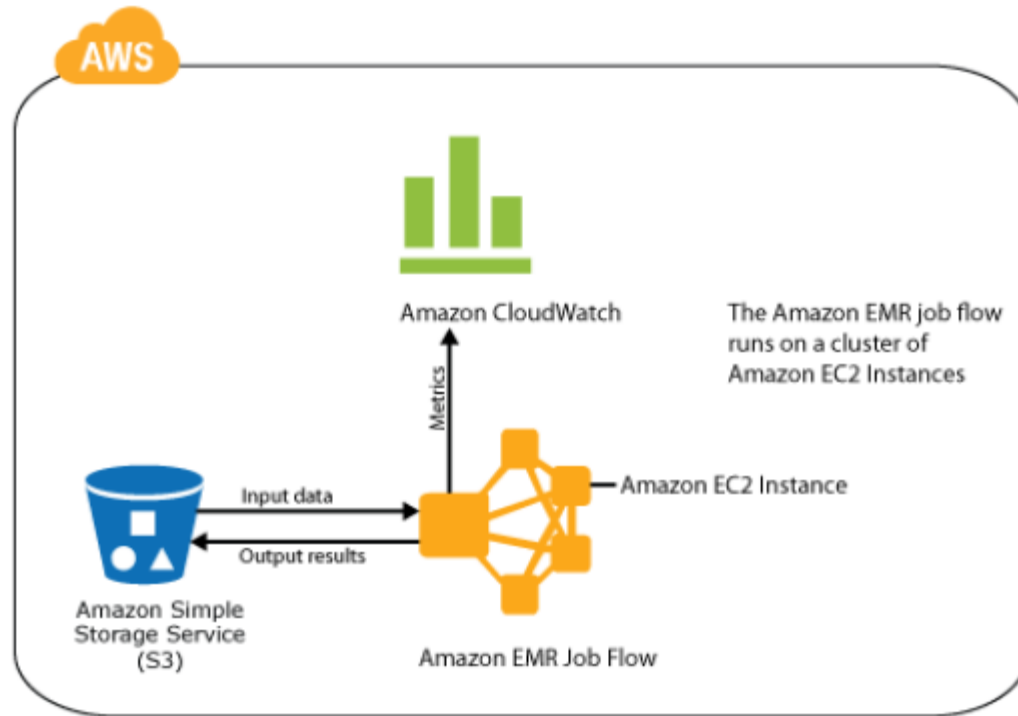
With Amazon EC2



On Amazon: EC2 and S3



AWS EMR Automation

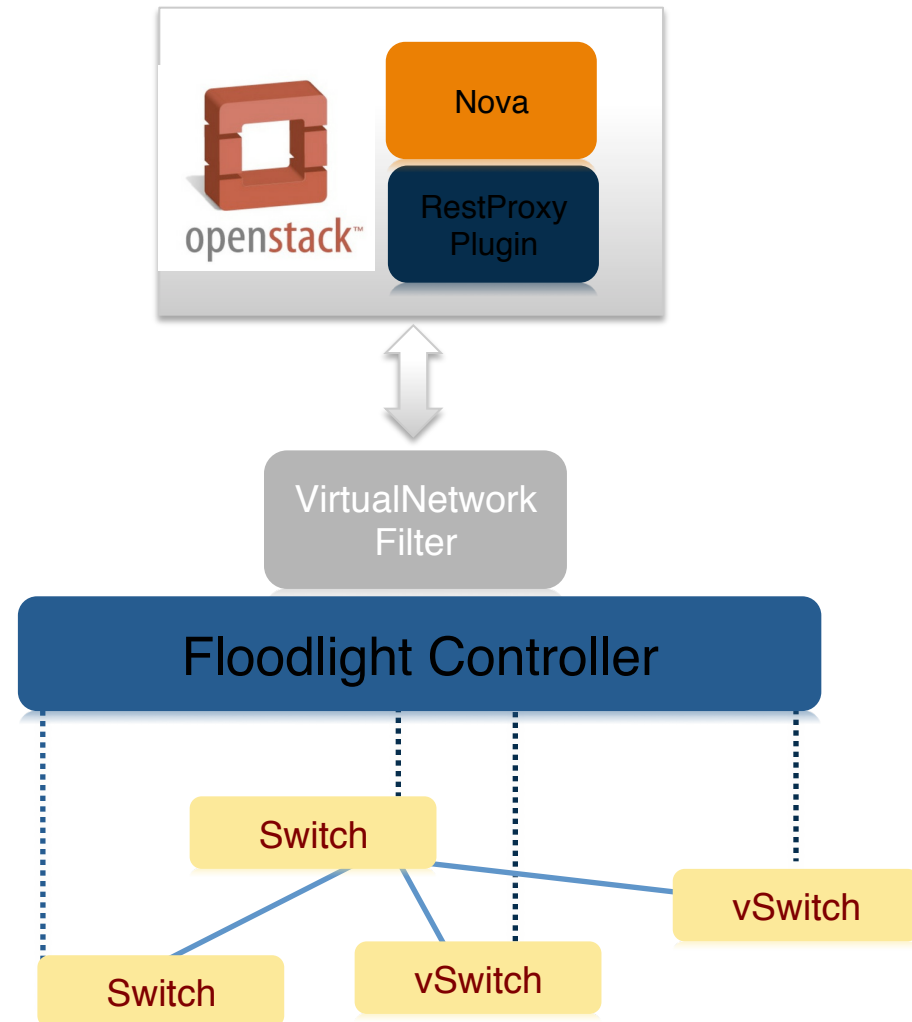


- Data input and output using S3
- Hadoop cluster based on EC2 – made auto-scaling leveraging CloudWatch and other services
- Data flow between S3 and HDFS is transparent

Part 3: Cloud Network Enablement: Software Defined Networking

Networking in OpenStack

- Nova: Network Orchestration Module
 - Can be used to create virtual networks for a group of VMs, tenants
 - Utilizes SDN Controllers
 - Floodlight
 - OpenDayList.



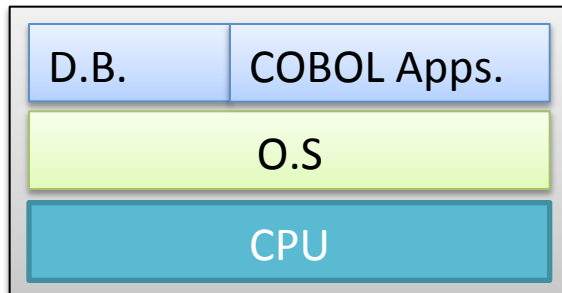
Outline

- Section 1: SDN Ecosystem
 - SDN Motivation
 - SDN Primer
 - Dimensions of SDN Environments
 - Dimensions of SDN Applications
- Section 2: OpenFlow Primer
- Section 3: Demo/Use-cases
 - Network Virtualization
- Section 4: SDN Challenges
 - SDN Challenges

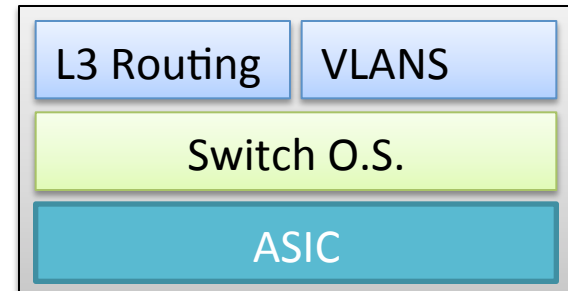
Section 1

Network Today...

- Vertical integrated stacks
 - Similar to PC in 1980s



IBM's Mainframe



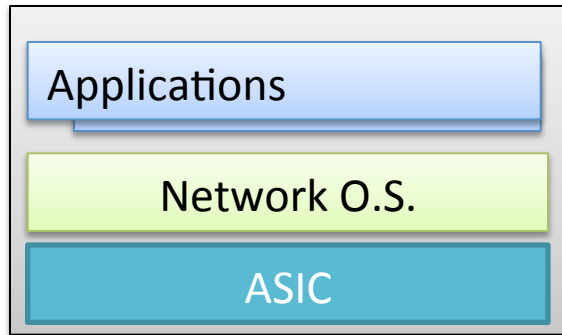
Cisco Routers

Implications of Networking...

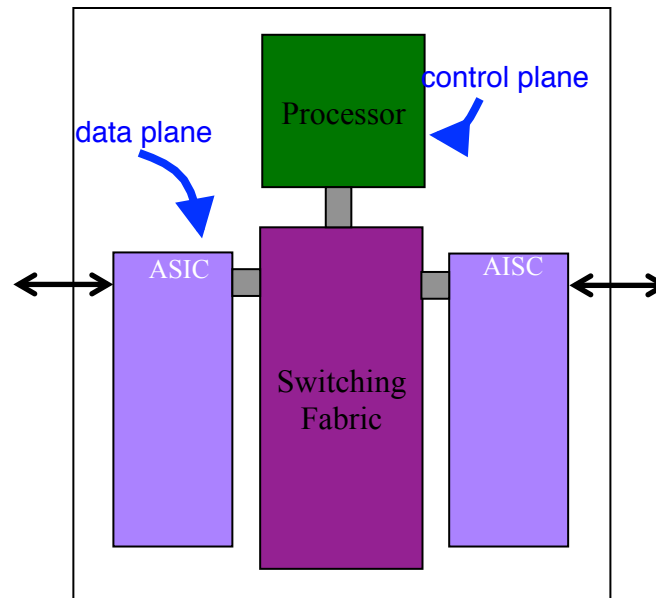
- Restricted to ill defined vendor CLI
 - Provisioning is slow....
 - VM provisioning: 1min
 - Virtual network provisioning: 1-3 weeks

Background: Switch Internals

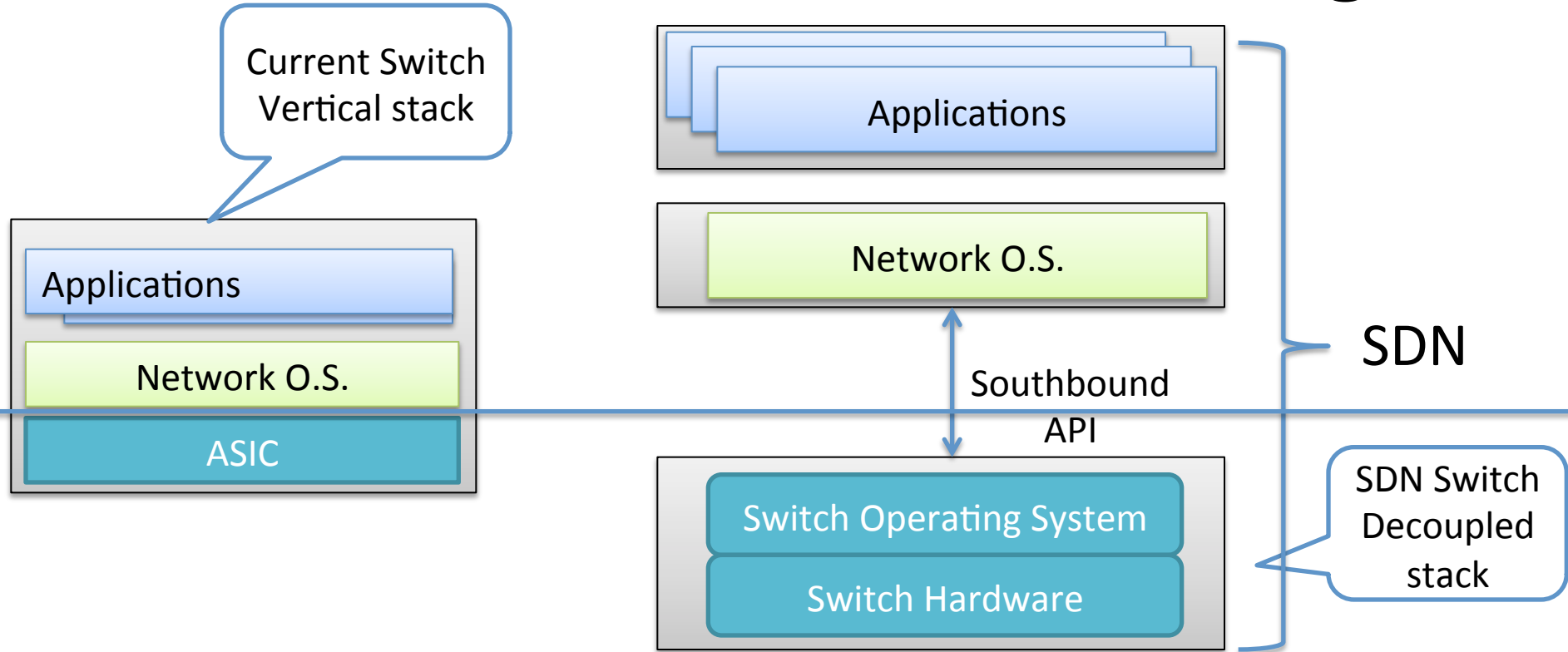
Logical View of a Switch



Physical Architecture of a Switch



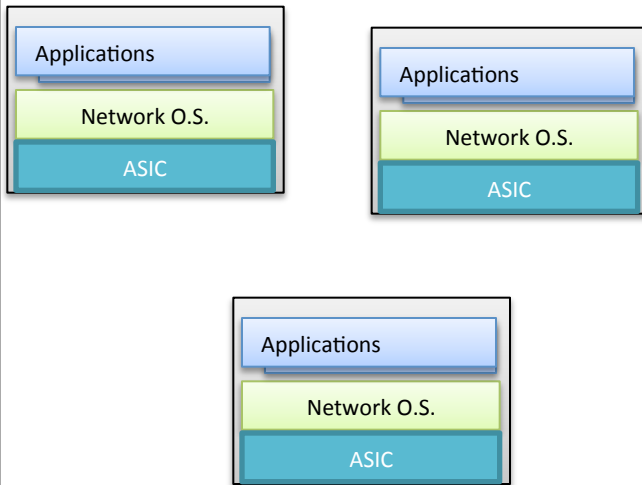
Software Defined Networking



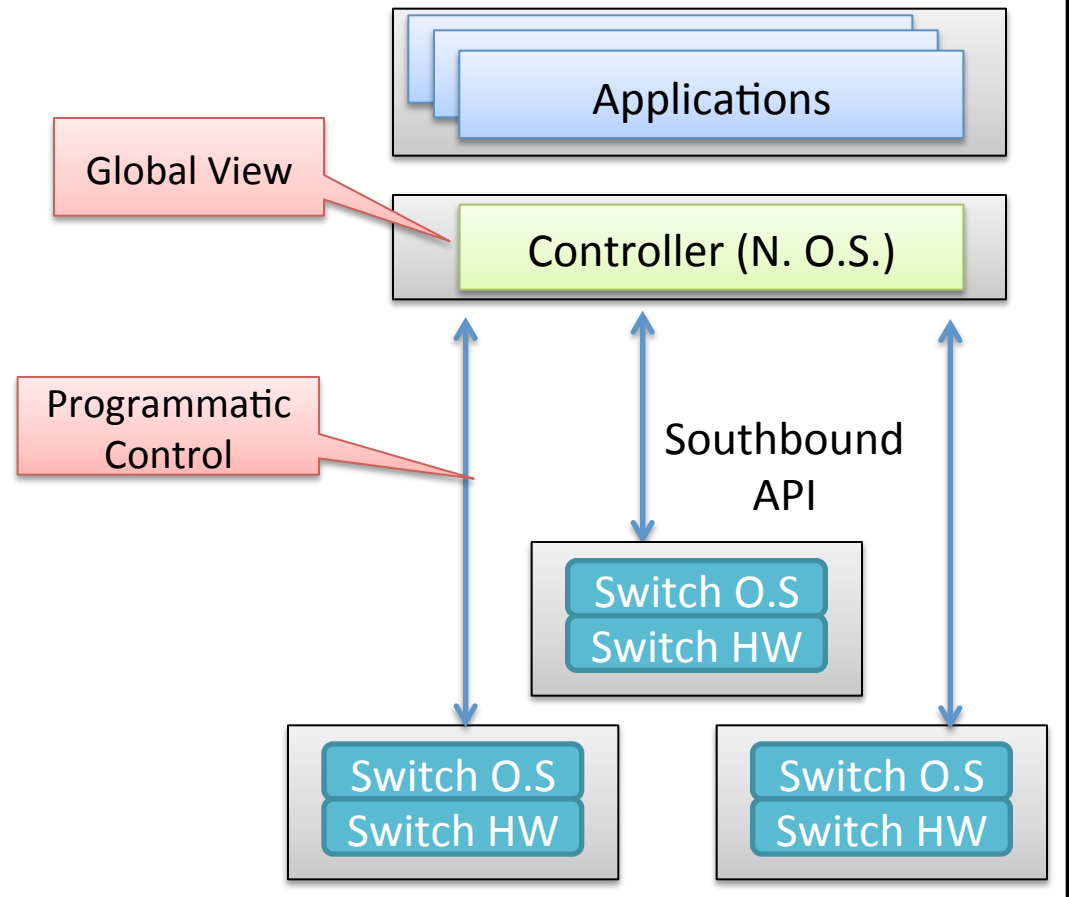
- Southbound API: decouples the switch hardware from control function
 - Data plane from control plane
- Switch Operating System: exposes switch hardware primitives

Implications Of SDN

Current Networking

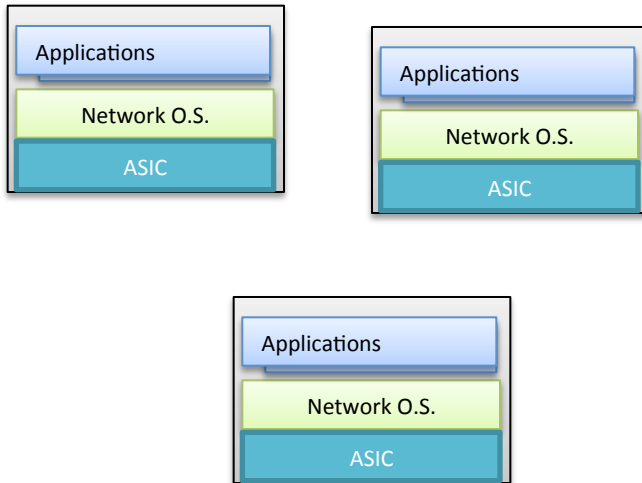


SDN Enabled Environment



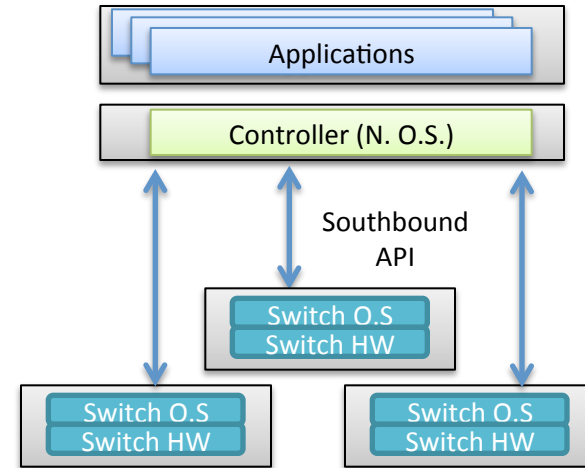
Implications Of SDN

Current Networking



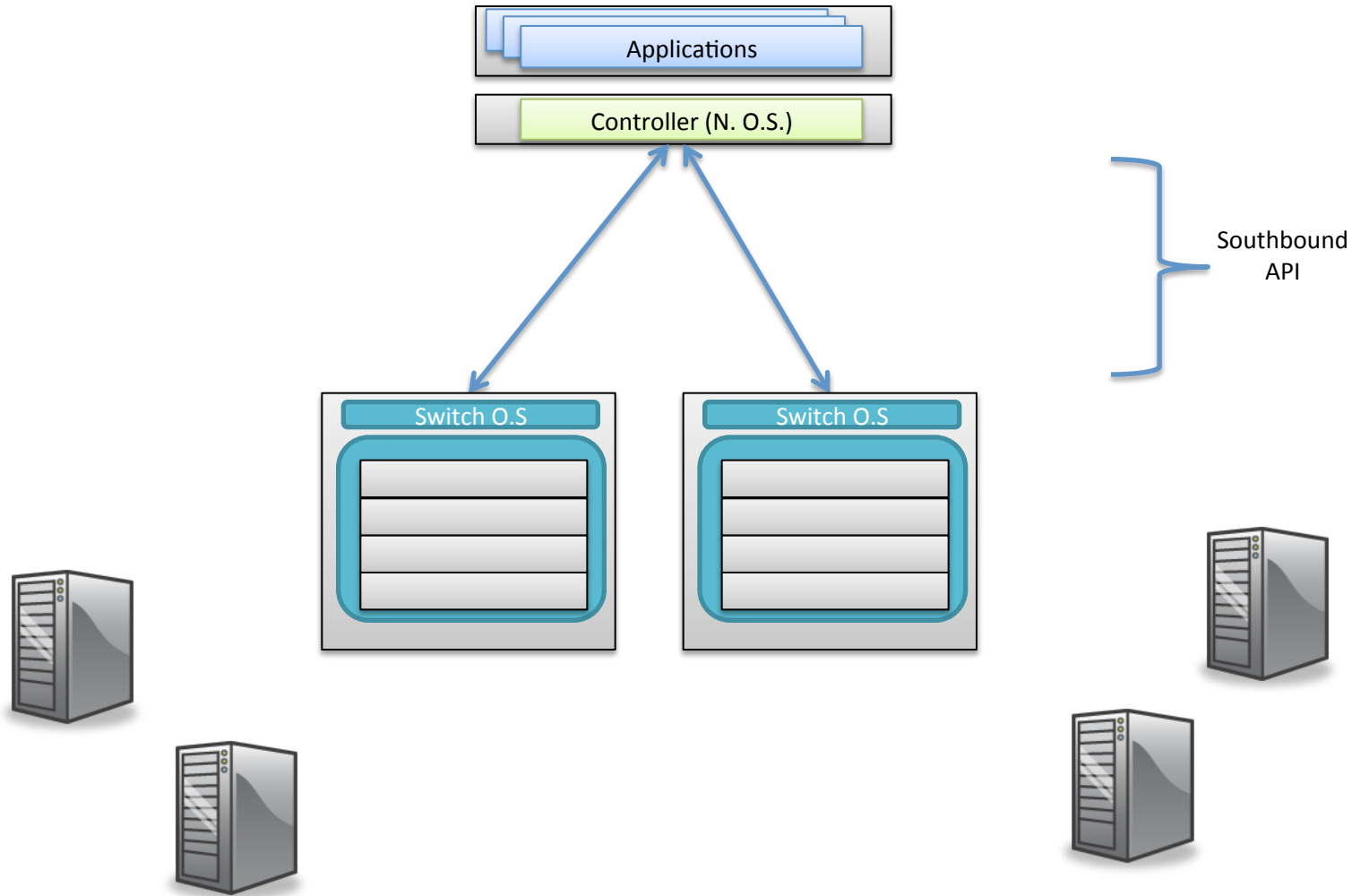
- Distributed protocols
 - Each switch has a brain
 - Hard to achieve optimal solution
- Network configured indirectly
 - Configure protocols
 - Hope protocols converge

SDN Enabled Environment



- Global view of the network
 - Applications can achieve optimal
- Southbound API gives fine grained control over switch
 - Network configured directly
 - Allows automation
 - Allows definition of new interfaces

How SDN Works



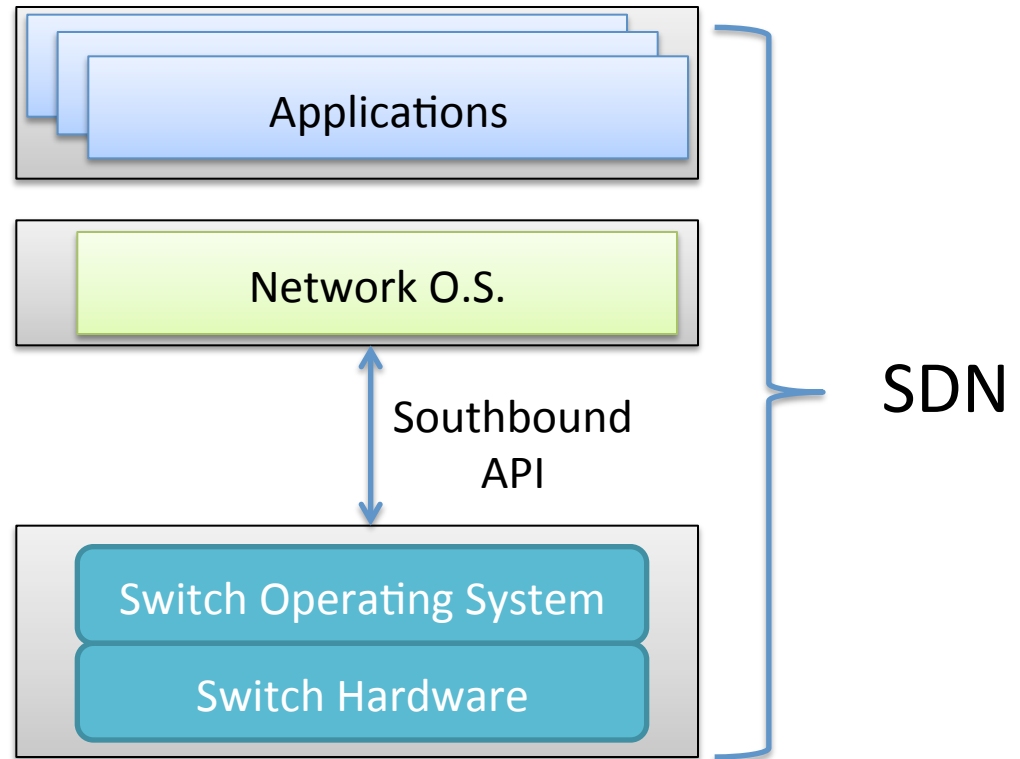
How to Pick an SDN Environment

How easy is it to develop
on for the
Controller platform?

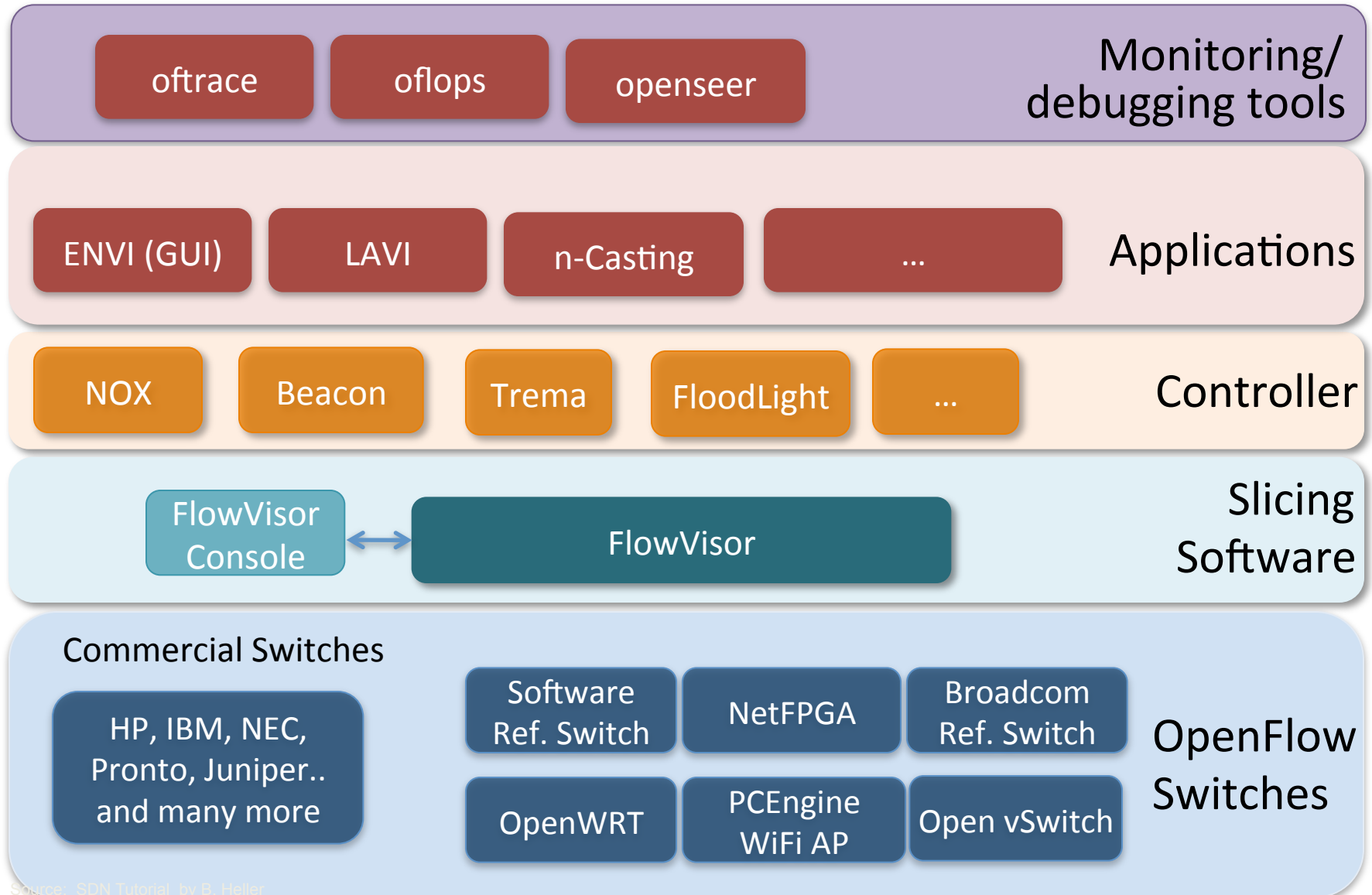
What is the Southbound API!?

Is the switch virtual or
physical?

Is the switch hardware
and OS closed?



The SDN Stack



Dimensions of SDN Environments:

Vendor Devices

Vertical Stacks

- Vendor bundles switch and switch OS
 - Restricted to vendor OS and vendor interface
- Low operational overhead
 - One stop shop

Whitebox Networking

- Vendor provides hardware with no switch OS
- Switch OS provided by third party
 - Flexibility in picking OS
- High operational overhead
 - Must deal with multiple vendors

Dimensions of SDN Environments:

Switch Hardware

Virtual: Overlay

- Pure software implementation
 - Assumes programmable virtual switches
 - Run in Hypervisor or in the OS
 - Larger Flow Table entries (more memory and CPU)
- Backward compatible
 - Physical switches run traditional protocols
- Traffic sent in tunnels
 - Lack of visibility into physical network

Physical: Underlay

- Fine grained control and visibility into network
- Assumes specialized hardware
 - Limited Flow Table entries

Dimensions of SDN Environments:

Southbound Interface

OpenFlow

- Flexible matching
 - L2, L3, VLAN, MPLS
- Flexible actions
 - Encapsulation: IP-in-IP
 - Address rewriting:
 - IP address
 - Mac address

BGP/XMPP/IS-IS/NetConf

- Limited matching
 - IS-IS: L3
 - BGP+MPLS: L3+MPLS
- Limited actions
 - L3/l2 forwarding
 - Encapsulation

Dimensions of SDN Environments:

Controller Types

Modular Controllers

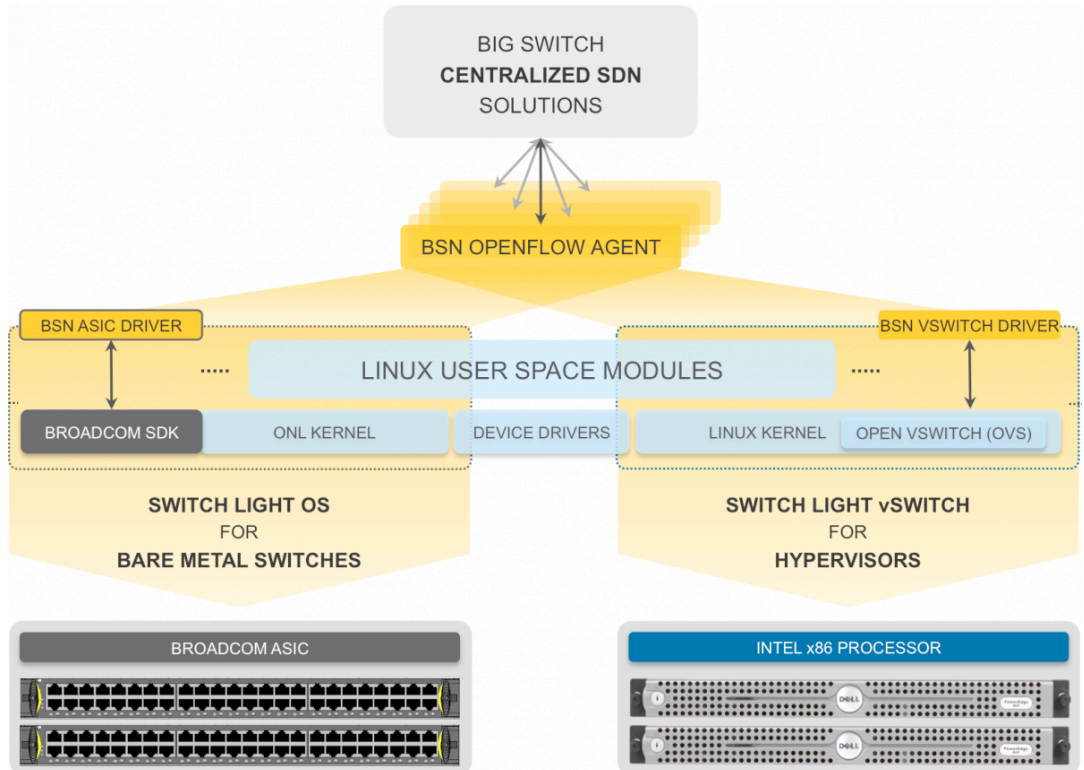
- Application code manipulates forwarding rules
 - E.g. OpenDaylight, Floodlight
- Written in imperative languages
 - Java, C++, Python
- Dominant controller style

High Level Controllers

- Application code specifies declarative policies
 - E.g. Frenetic, McNettle
- Application code is verifiable
 - Amendable to formal verification
- Written in functional languages
 - Nettle, OCamal

BigSwitch

- Controller Type
 - Modular: Floodlight
- Southbound API: OpenFlow
 - OpenFlow 1.3
- SDN Device: Whitebox
 - (indigo)
- SDN Flavor
 - Underlay+Overlay



Juniper Contrail

- Controller Type
 - Modular: OpenContrail
- Southbound API: XMPP/NetConf
 - BGP+MPLS
- SDN Device: Vertical Stack
 - Propriety Junos
- SDN Flavor
 - Overlay

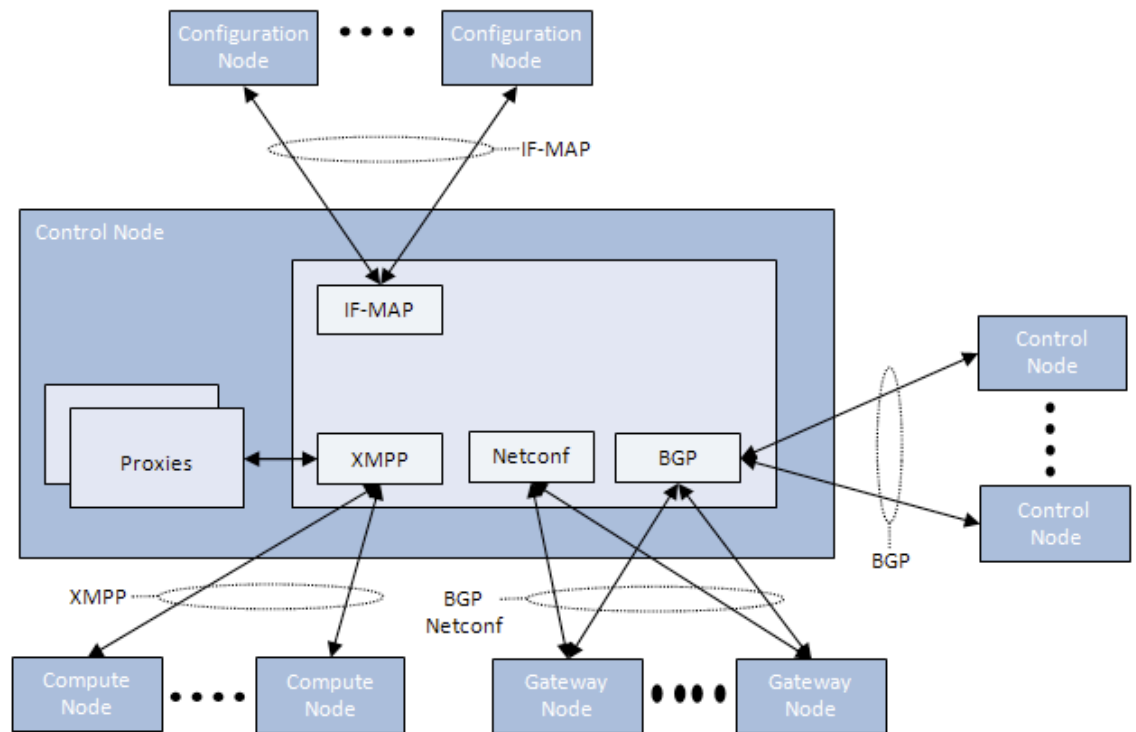
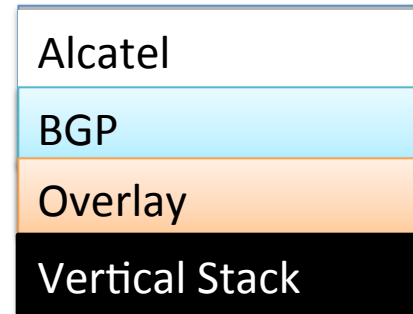
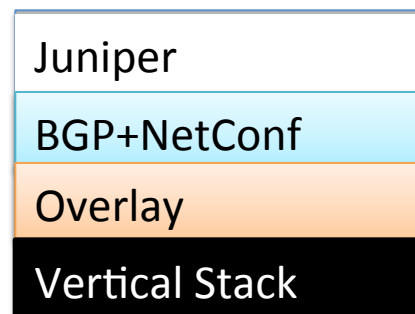
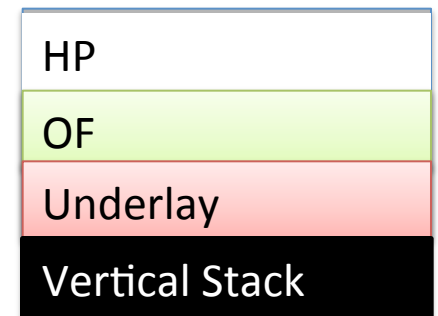
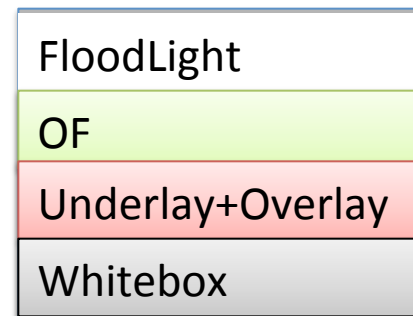
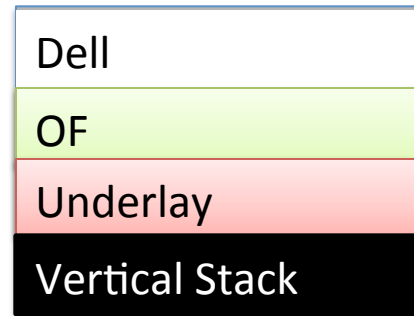
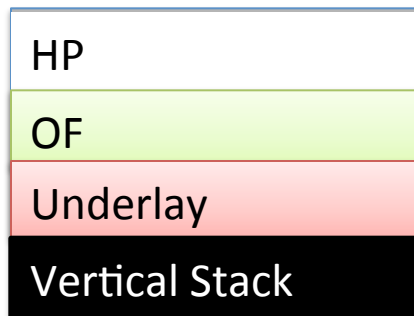
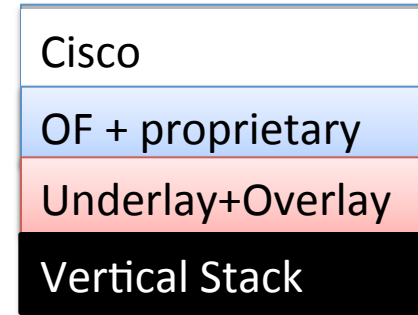
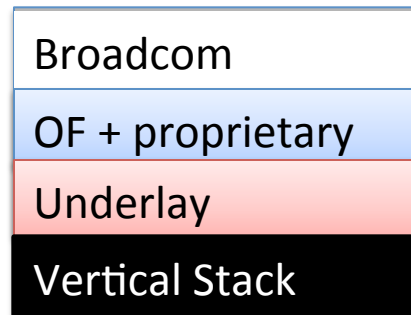
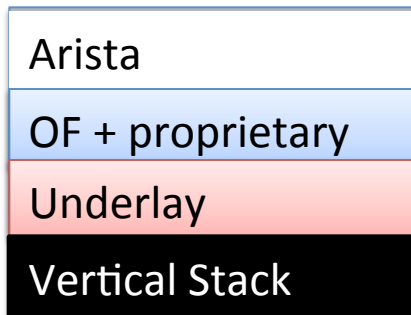
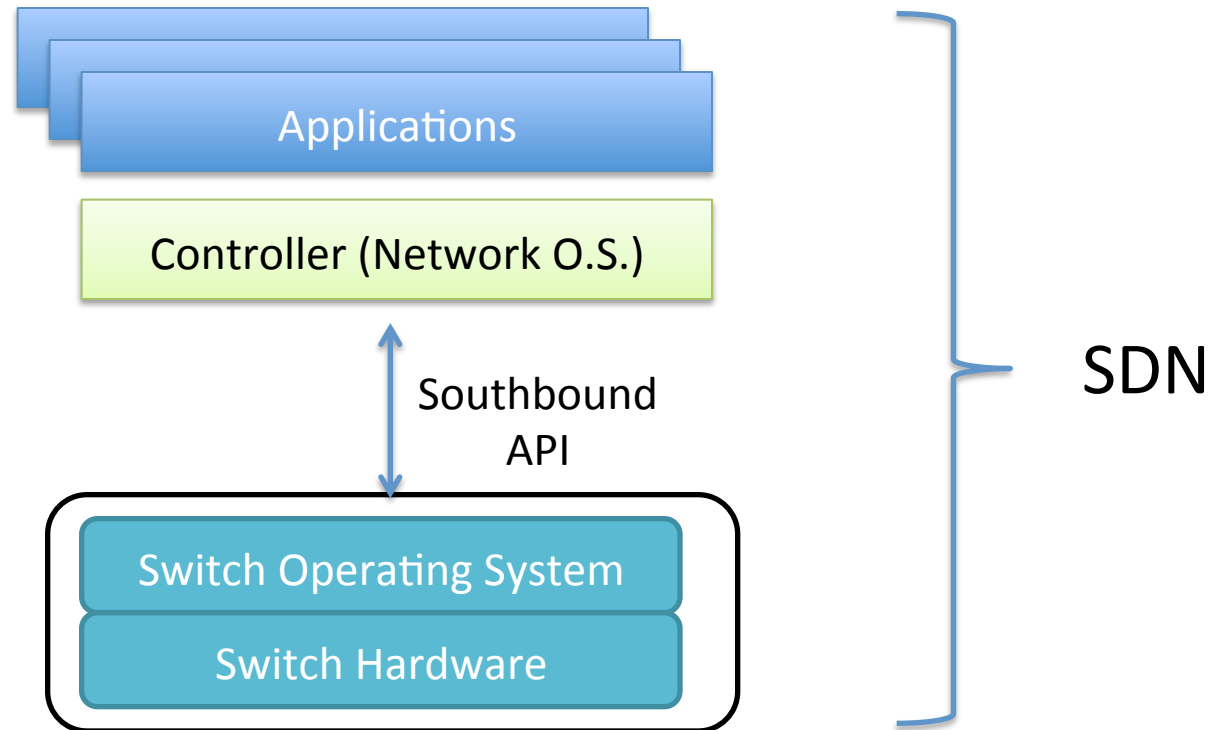


Figure 5: Internal Structure of a Control Node

SDN EcoSystem



SDN Stack



- Southbound API: decouples the switch hardware from control function
 - Data plane from control plane
- Switch Operating System: exposes switch hardware primitives

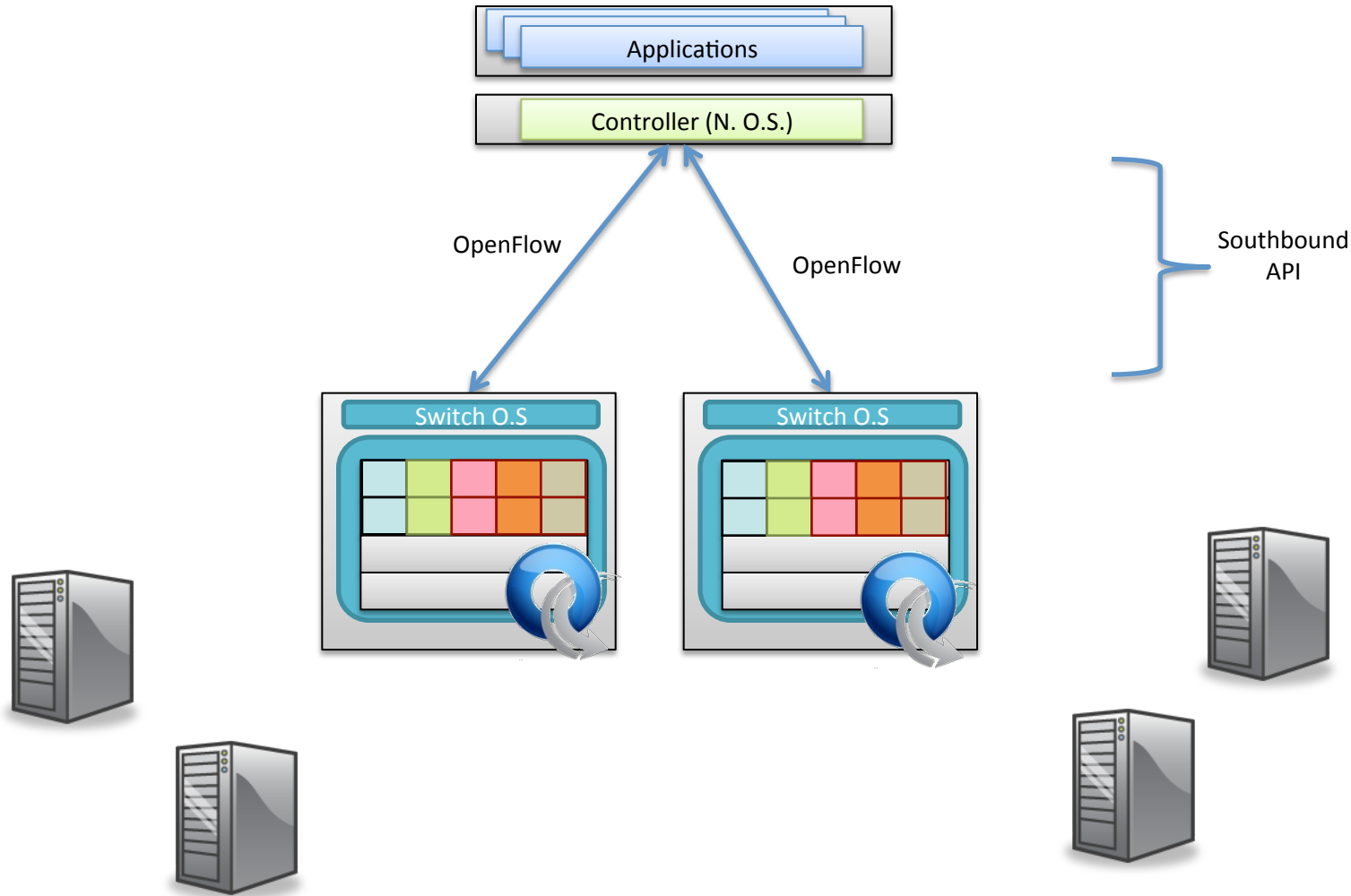
Section2: Southbound API: OpenFlow

OpenFlow

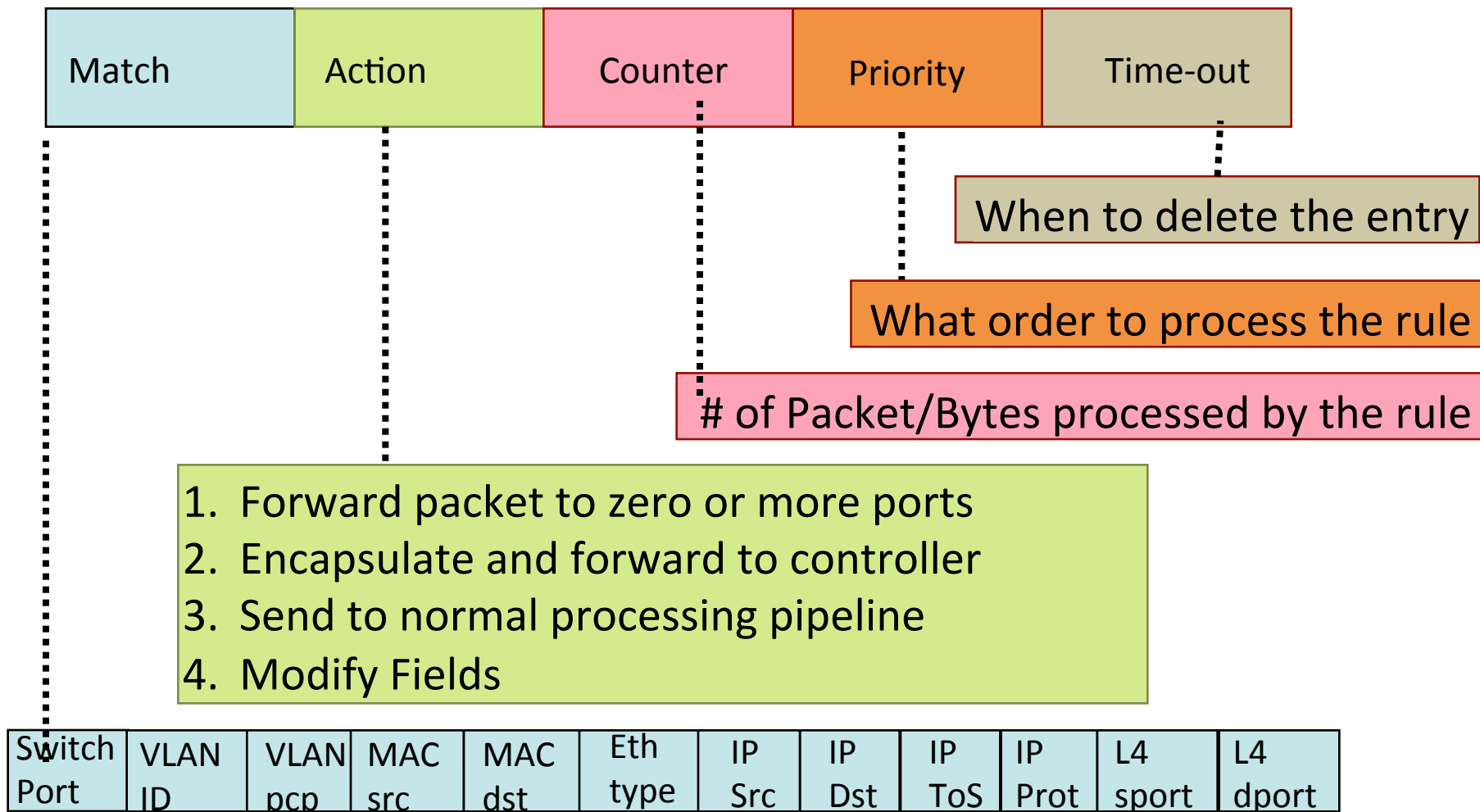
- Developed in Stanford
 - Standardized by Open Networking Foundation (ONF)
 - Current Version 1.4
 - Version implemented by switch vendors: 1.3
- Allows control of underlay + overlay
 - Overlay switches: OpenVSwitch/Indigo-light



How SDN Works: OpenFlow



OpenFlow: Anatomy of a Flow Table Entry

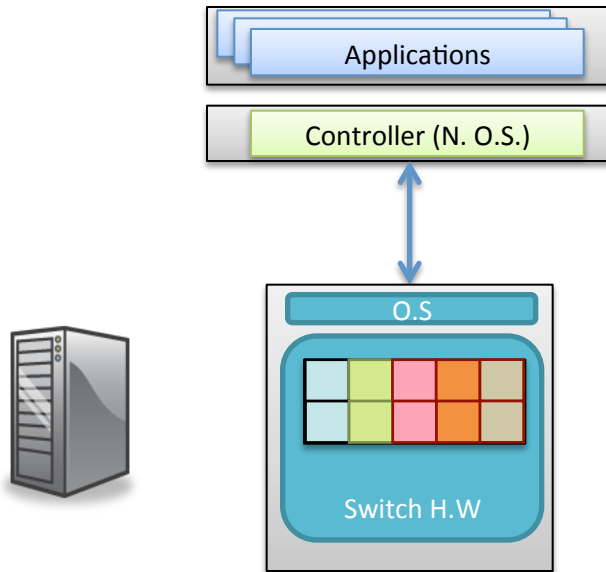


OpenFlow: Types of Messages

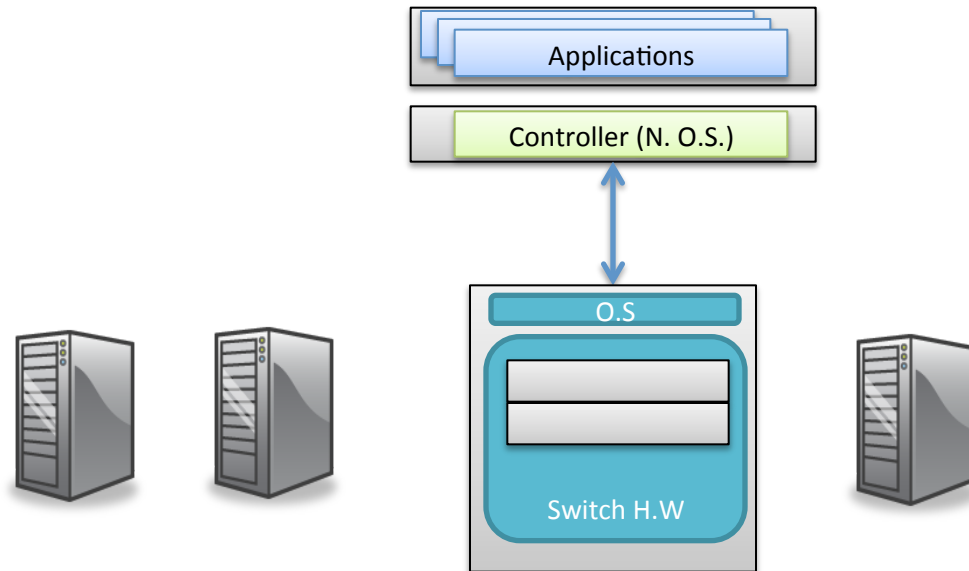
- Asynchronous (Controller-to-Switch)
 - **Send-packet:** to send packet out of a specific port on a switch
 - **Flow-mod:** to add/delete/modify flows in the flow table
- Asynchronous (initiated by the switch)
 - **Read-state:** to collect statistics about flow table, ports and individual flows
 - **Features:** sent by controller when a switch connects to find out the features supported by a switch
 - **Configuration:** to set and query configuration parameters in the switch
- Asynchronous (initiated by the switch)
 - **Packet-in:** for all packets that do not have a matching rule, this event is sent to controller
 - **Flow-removed:** whenever a flow rule expires, the controller is sent a flow-removed message
 - **Port-status:** whenever a port configuration or state changes, a message is sent to controller
 - **Error:** error messages
- Symmetric (can be sent in either direction without solicitation)
 - **Hello:** at connection startup
 - **Echo:** to indicate latency, bandwidth or liveliness of a controller-switch connection
 - **Vendor:** for extensions (that can be included in later OpenFlow versions)

Dimension of SDN Applications: Rule installation

Proactive Rules



Reactive Rules



Dimension of SDN Applications:

Rule installation

Proactive Rules

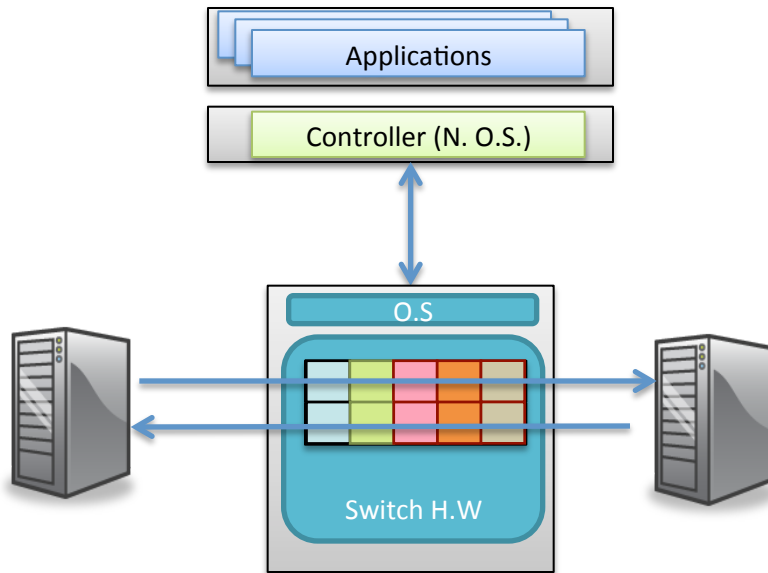
- Controller pre-installs flow table entries
 - Zero flow setup time
- Requires installation of rules for all possible traffic patterns
 - Requires use of aggregate rules (Wildcards)
 - Require foreknowledge of traffic patterns
 - Waste flow table entries

Reactive Rules

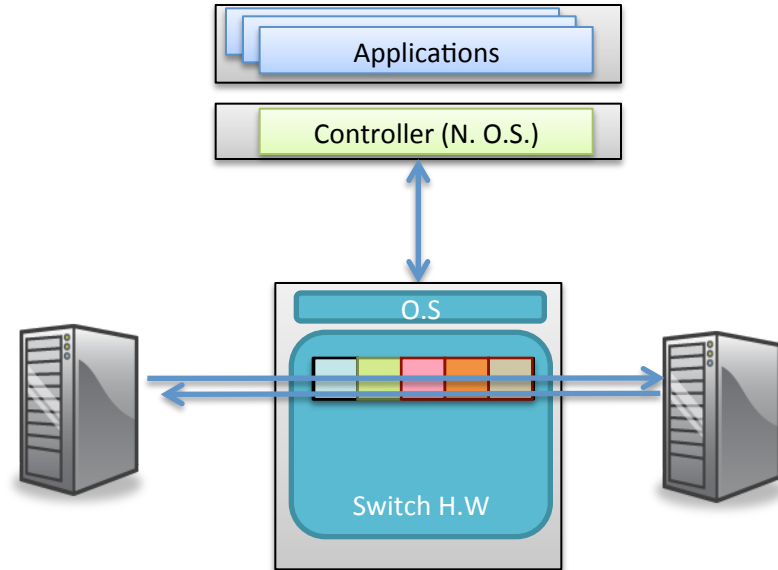
- First packet of each flow triggers rule insertion by the controller
 - Each flow incurs flow setup time
 - Controller is bottleneck
 - Efficient use of flow tables

Dimensions of SDN Applications: Granularity of Rules

Microflow



WildCards (aggregated rules)



Dimensions of SDN Applications:

Granularity of Rules

Microflow

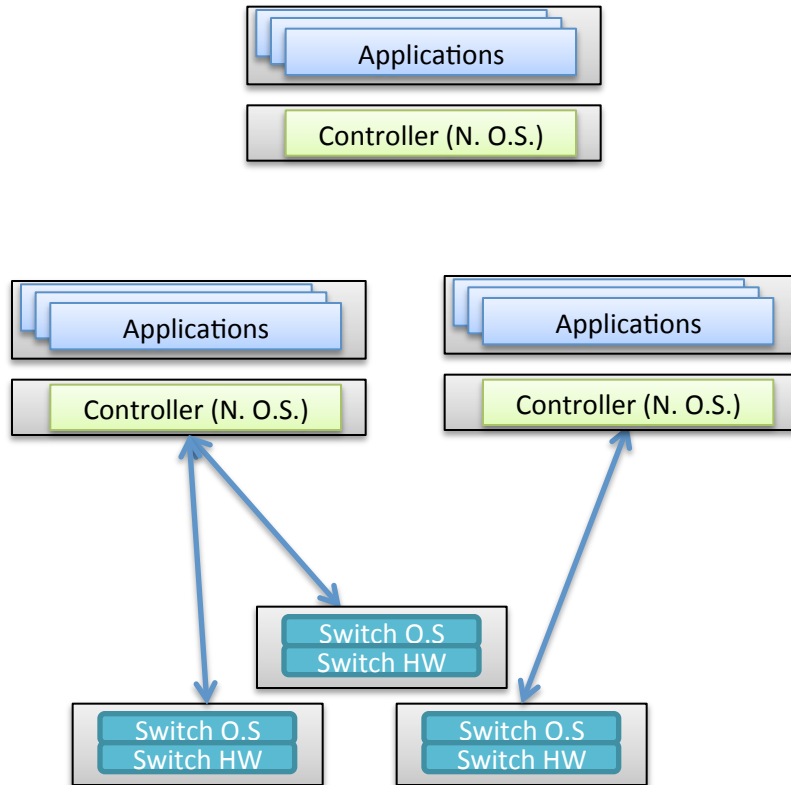
- One flow table matches one flow
- Uses CAM/hash-table
 - 10-20K per physical switch
- Allows precisions
 - Monitoring: gives counters for individual flows
 - Access-Control: allow/deny individual flows

WildCards (aggregated rules)

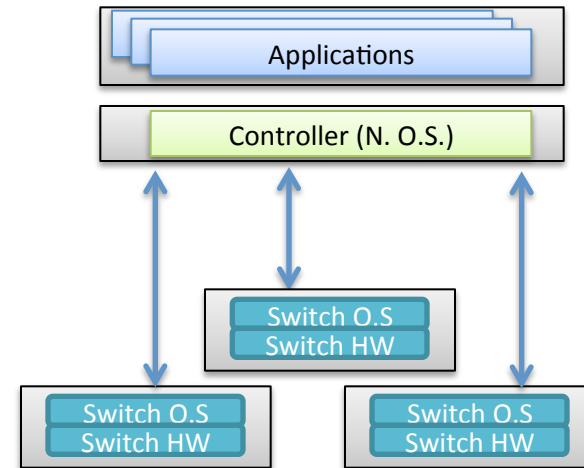
- One flow table entry matches a group of flow
- Uses TCAM
 - 5000~4K per physical switch
- Allows scale
 - Minimizes overhead by grouping flows

Dimensions of SDN Applications: Granularity of Rules

Distributed Controller



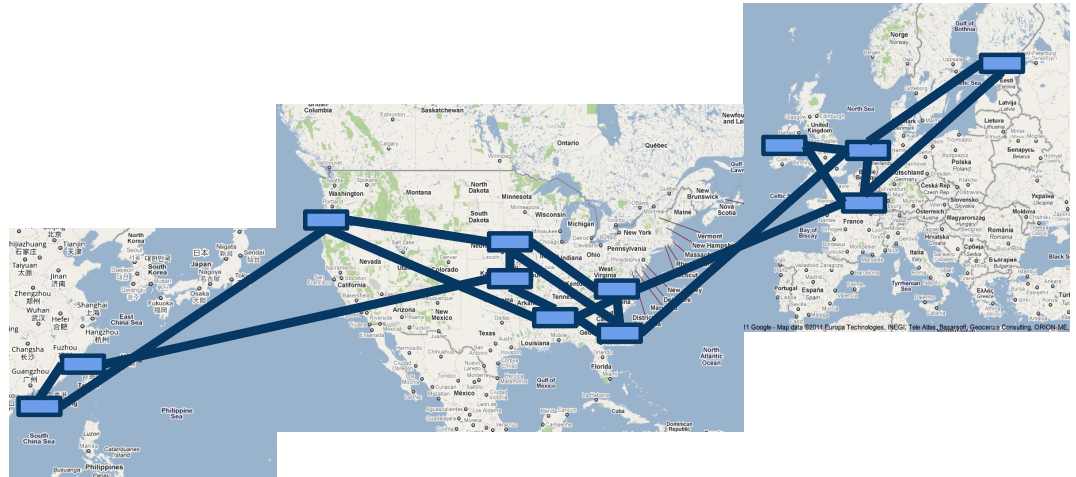
Centralized Controller



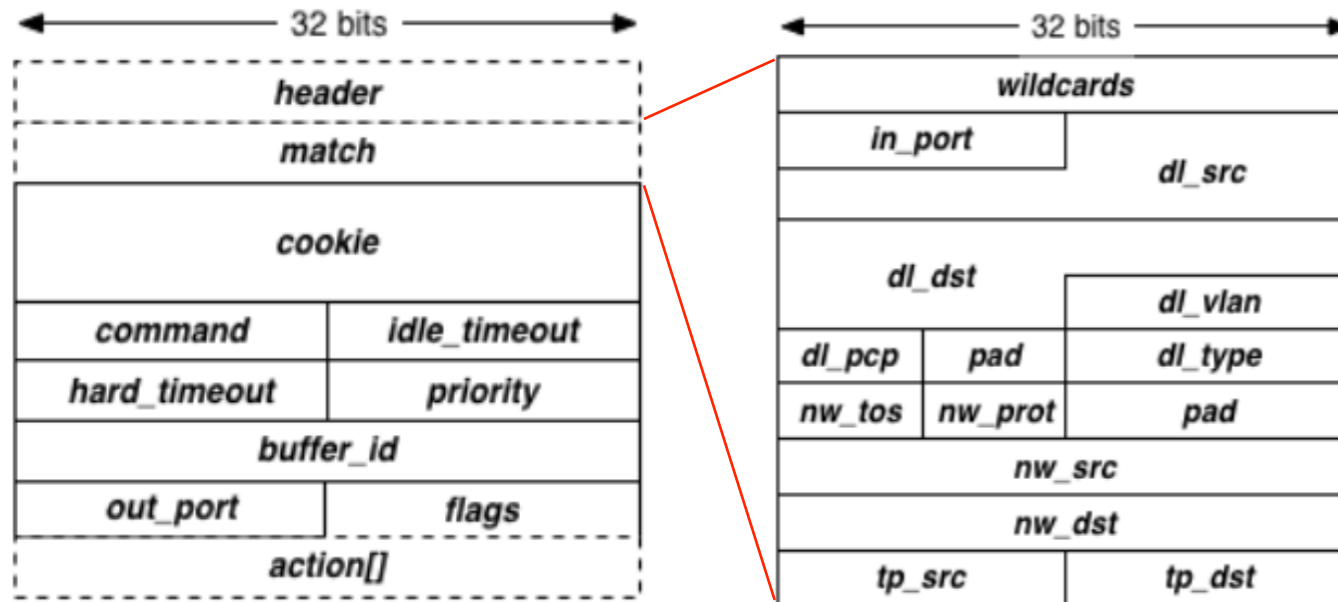
Google's Software Defined WANGoogle

Google' B4 Application

- Rule installation
 - Proactive
- Rule Granularity
 - Aggregate
- Distributed
 - Multiple instances



OpenFlow: Message Formats



- Controller encapsulates message into an object
 - Accessor functions to different fields
 - No need to worry about crafting network packets

OpenFlow Actions (Partial list from OpenFlow 1.0 spec)

- Output to switch port (Physical ports & virtual ports). Virtual ports include the following:
 - ALL (all standard ports excluding the ingress port) - flood
 - CONTROLLER (encapsulate and send the packet to controller) – PACKET_IN message
 - LOCAL (switch's stack) – go through the IP layer, etc (mostly used for vSwitches)
 - NORMAL (process the packet using traditional non-OpenFlow pipeline of the switch) – traditional L2 forwarding, L3 routing
- Drop
- Set fields (packet modification/header rewriting)
 - Ethernet Source address
 - Ethernet Dest address
 - IP source & dest addresses, IP ToS, IP ECN, IP TTL, VLAN
 - TCP/UDP source and destination ports
- Strip (pop) the outer VLAN tag
- Set queue ID when outputting to a port (Enqueue)
- New in OpenFlow 1.1+
 - Support for matching across multiple tables
 - Support for tunneling
 - Support for Push/Pop multiple VLAN/MPLS/PBB tags

Section 2: SDN Use Cases

SDN Use Cases

- Network Virtualization (VMWare, Azure)
- Port tapping (Big Switch's BigTap)
- Access control (Big Switch's SNAC)
- WAN Traffic Engineering (Google B4)
- DDoS Detection (Defense4All)
- Network Orchestration (OpenStack, VMWare)

SDN Use Cases

- WAN-Traffic engineering
 - Google's B4 (SIGCOMM 2013)
 - Microsoft's SWAN (SIGCOMM 2013)
- Network Function Virtualization: Service Chaining
 - SIMPLIFY/FlowTags (SIGCOMM 2013, NSDI 2014)
 - Slick (ONS 2013)
- Network virtualization
 - Nicira, Azure, Google,
 - VL2 & Portland (SIGCOMM 2009)
 - CloudNaaS (SoCC 2011)
- Seamless workload (VM) mobility
 - (CrossRoads (NOMS 2012))
- Data Center Traffic engineering
 - Routing elephant flows differently (Hedera – NSDI 2010)
 - Routing predictable traffic (MicroTE – CoNext 2011)
- Port-Mirroring
 - BigTap
 - OpenSafe (INM/WREN 2011)

SDN Use Case: Network Function Virtualization

Network Policy:

Web

Firewall

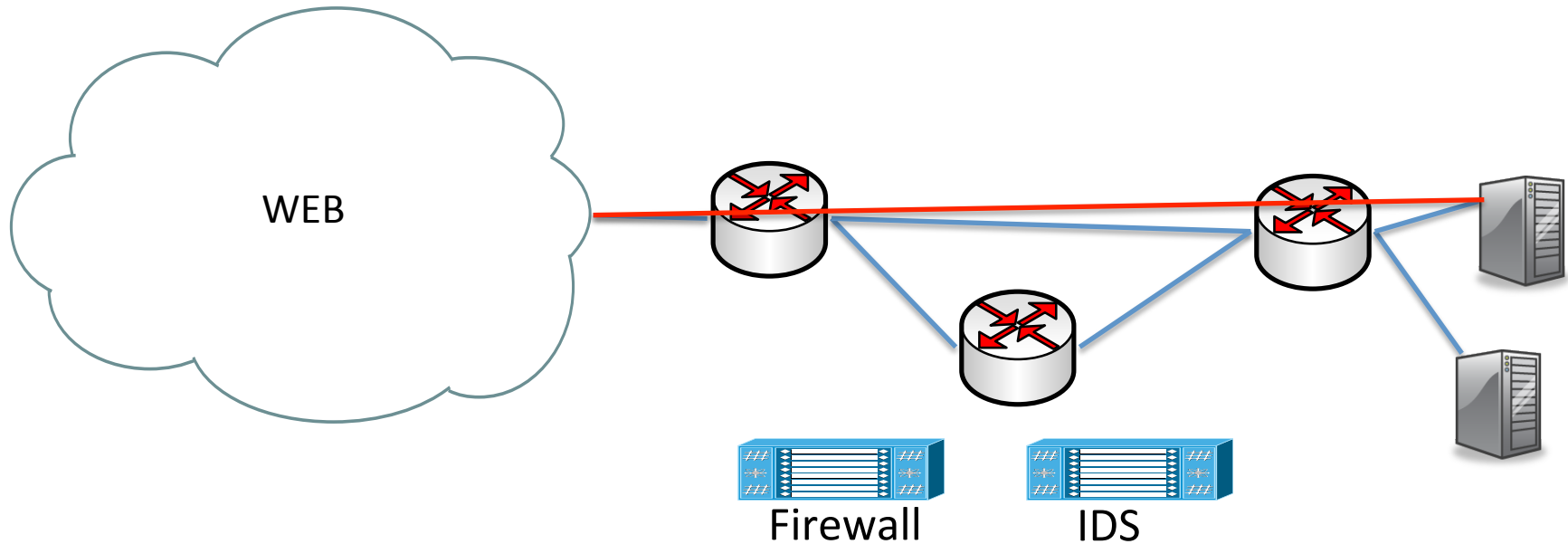
IDS

Goals:

- Detect attacks
- Prevent unauthorized access

Problem:

- Traffic takes shortest path
- Avoids middleboxes
 - Servers are unprotected

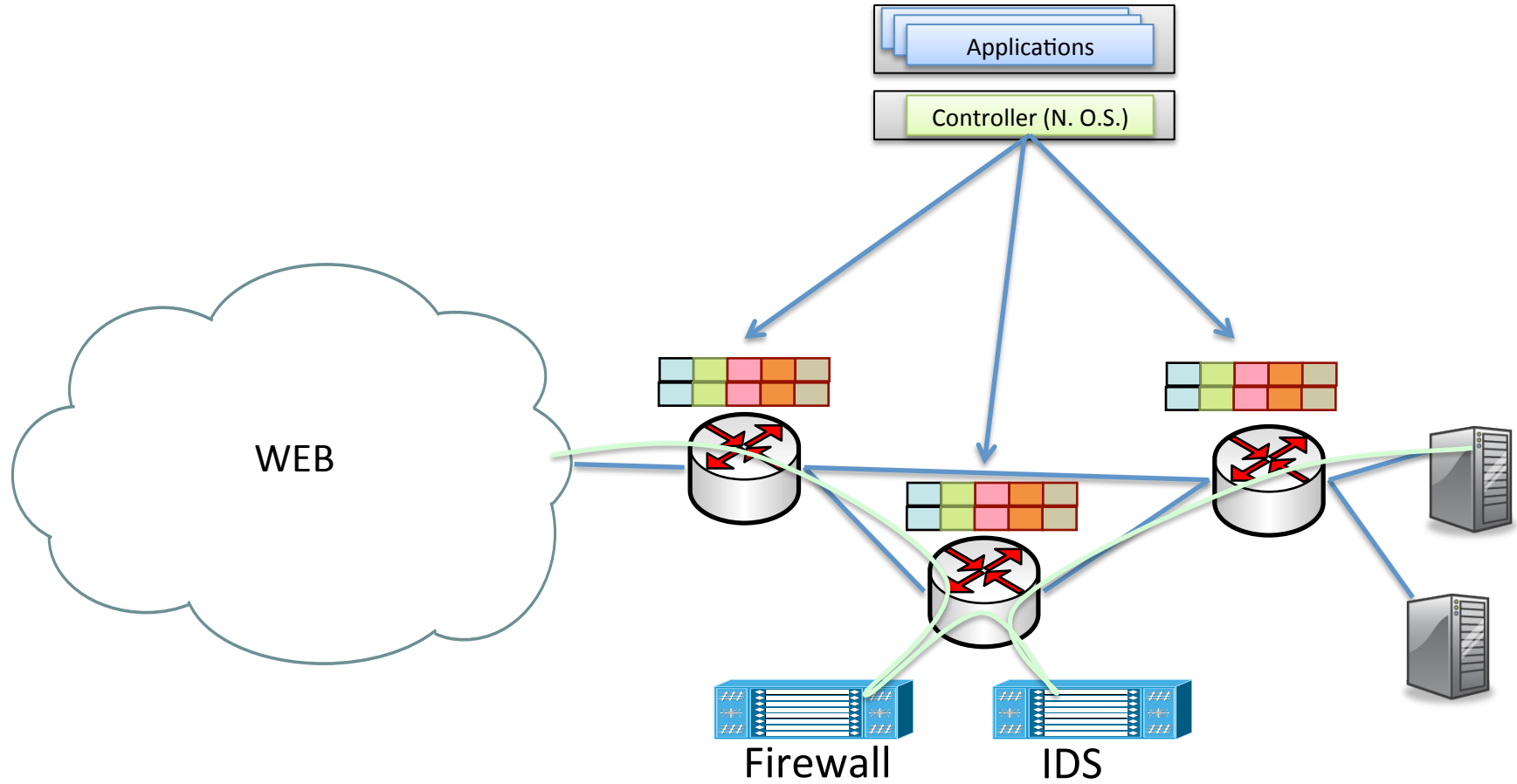


Network Policy:

Web

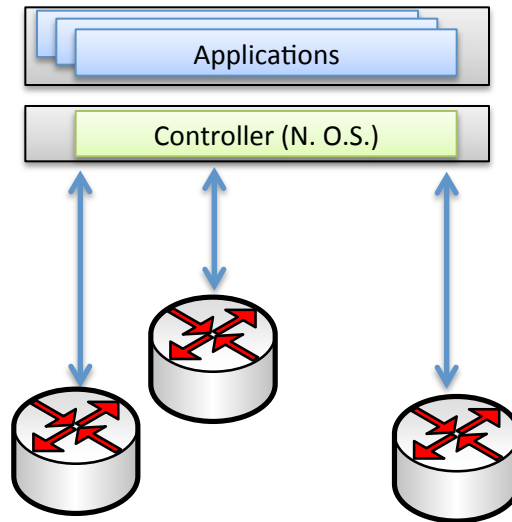
Firewall

IDS

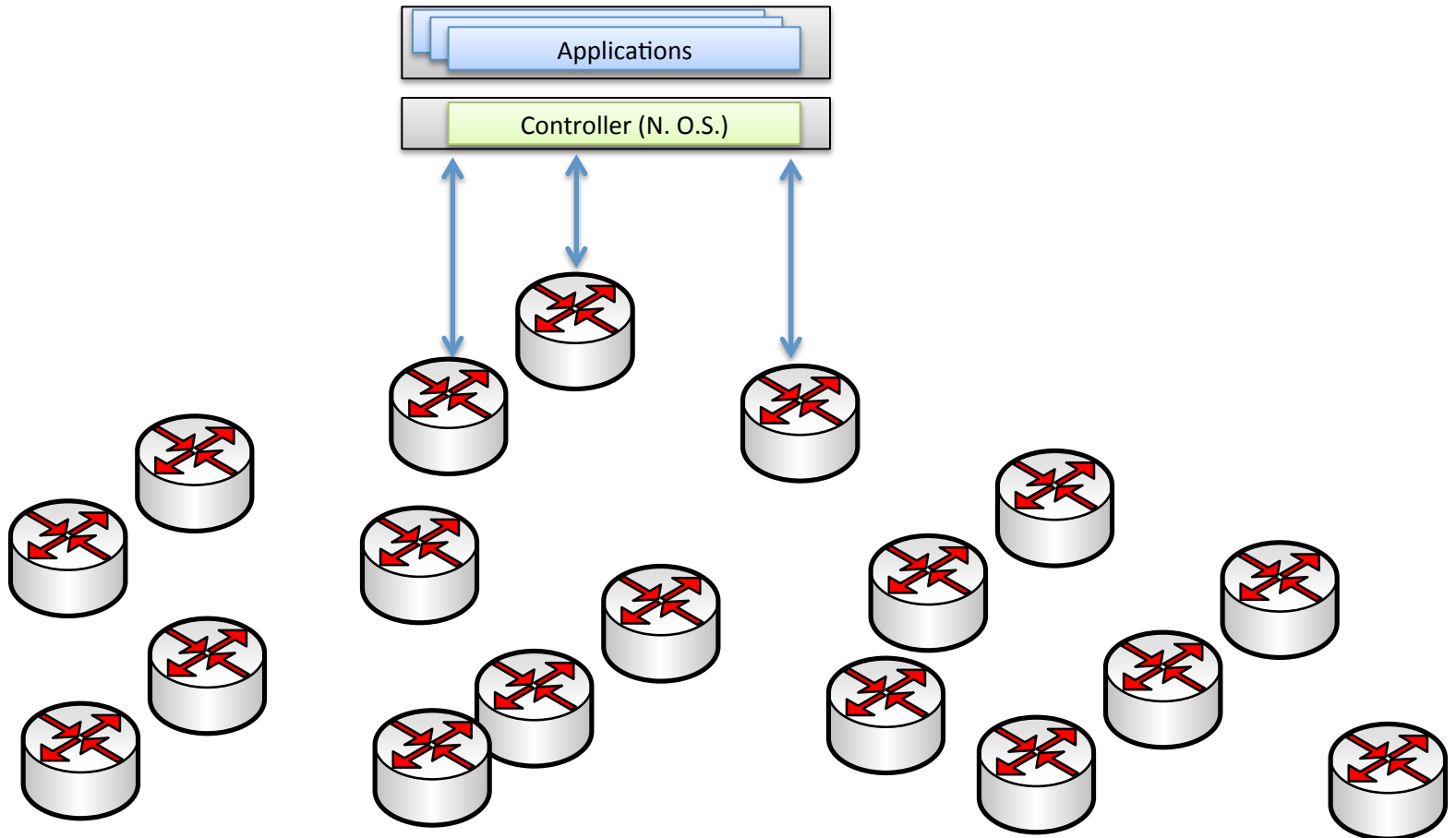


Section 3: SDN Challenges

Controller Availability



Controller Availability

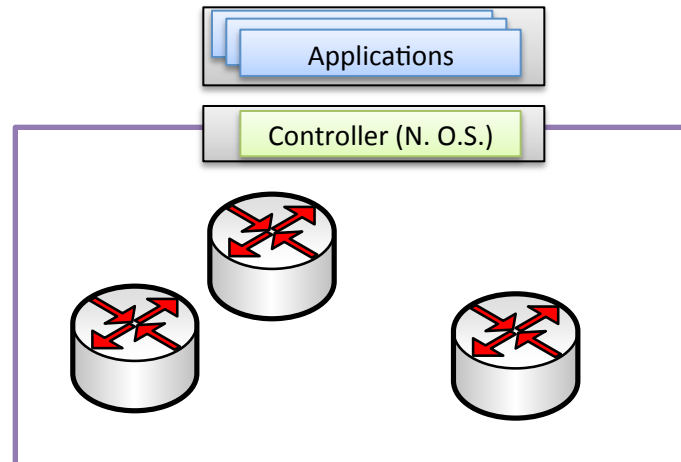
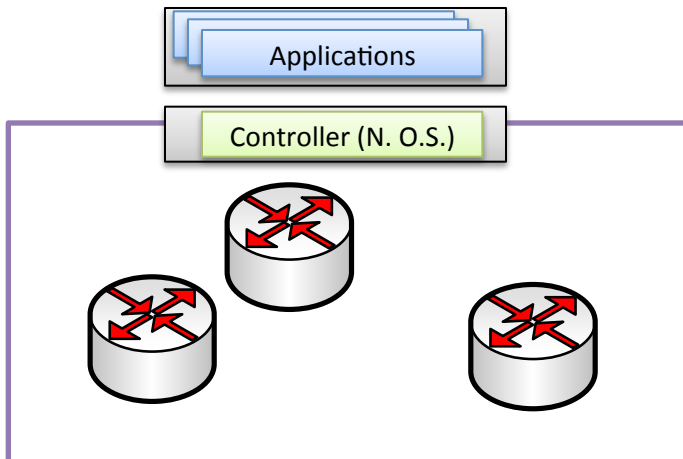
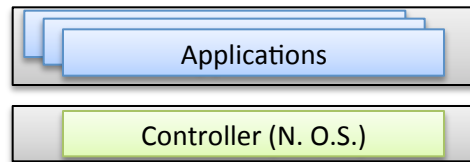


Controller Availability

“control a large force like a small force: divide and conquer”

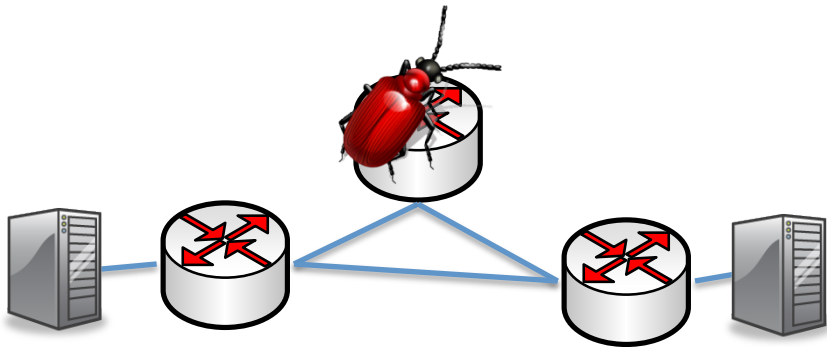
--Sun Tzu, Art of war

- How many controllers?
- How do you assign switches to controllers?
- More importantly: which assignment reduces processing time
- How to ensure consistency between controllers



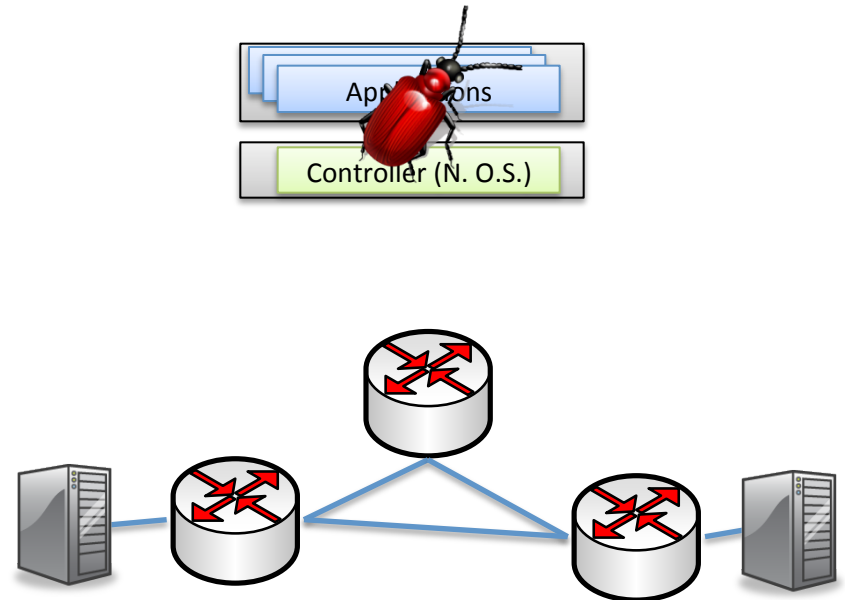
SDN Reliability/Fault Tolerance

Existing network survives failures or bugs in code for any one devices



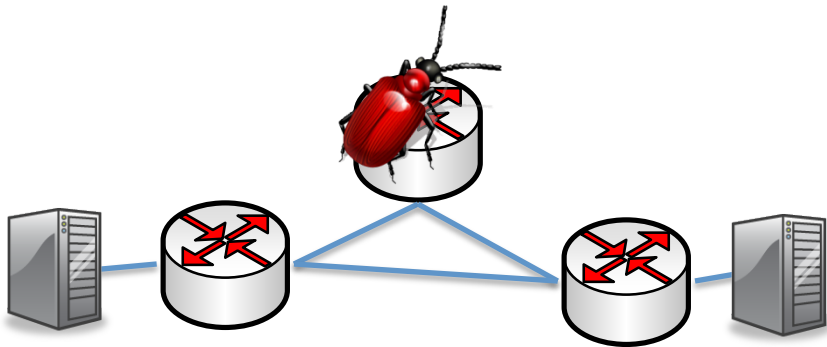
Controller: Single point of control

- Bug in controller takes the whole network down



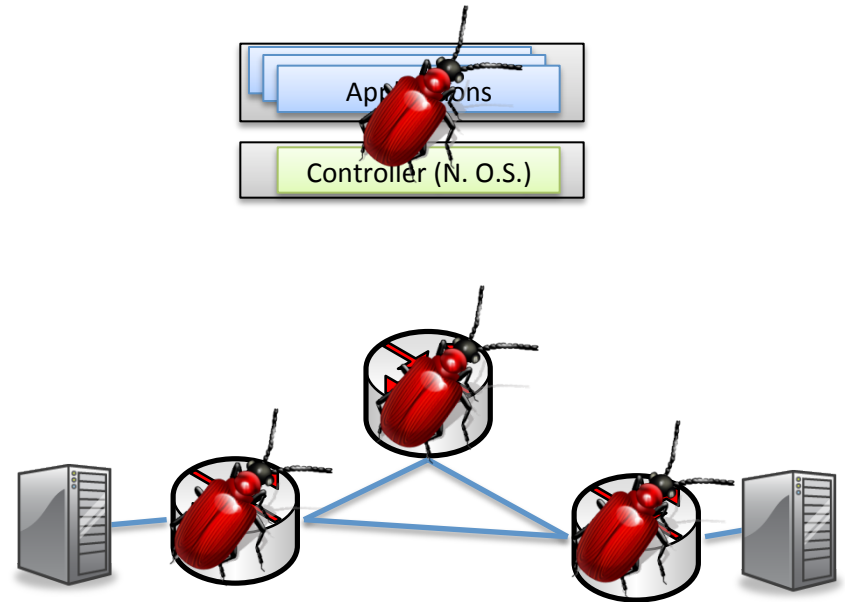
SDN Reliability/Fault Tolerance

Existing network survives failures or bugs in code for any one devices



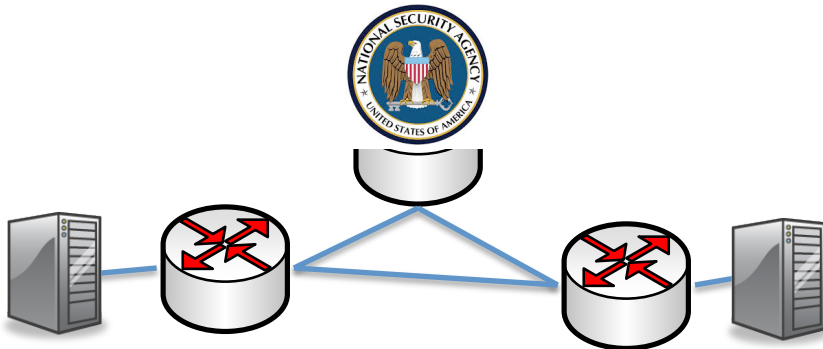
Controller: Single point of control

- Bug in controller takes the whole network down
- Single point of failure



SDN Security

If one device in the current networks are compromised the network may still be safe



Controller: Single point of control

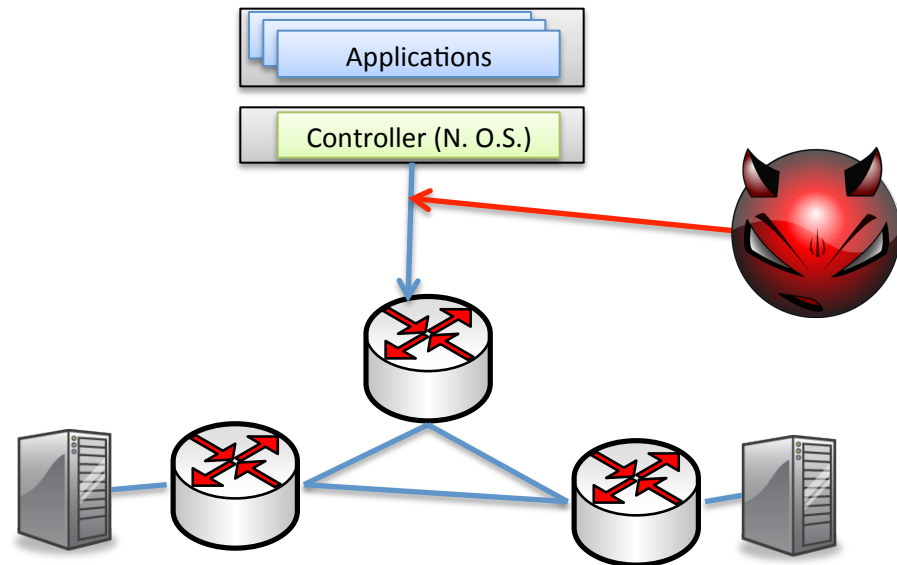
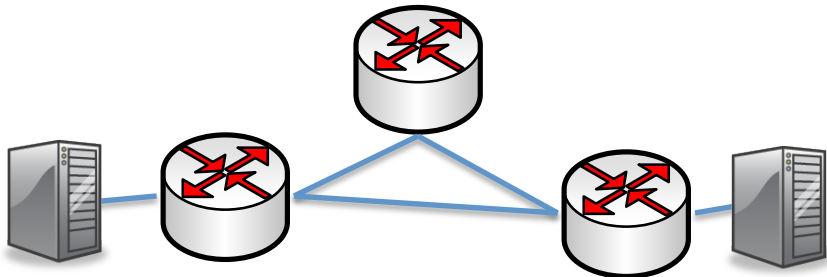
- Compromise controller



SDN Security

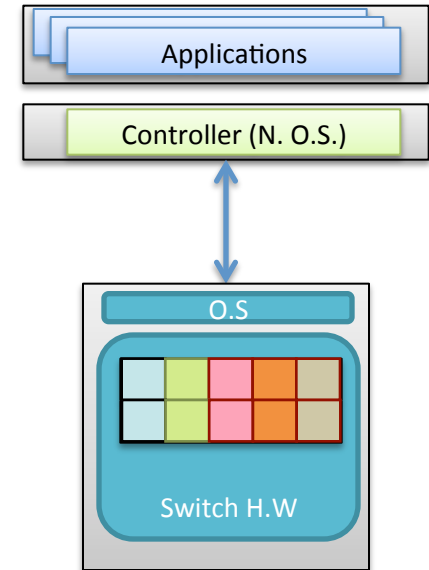
Controller: Single point of control

- Compromise controller
- Denial of Service attack the control channel



Data-Plane Limitations

- Limited Number of TCAM entries
 - How to fit network in limited entries?
- Limited control channel capacity
 - Need to rate limit control messages
- Limited switch CPU
 - Limit control messages and actions that use CPU



Conclusion

- An overview of SDN technologies
- Introduction to OpenFlow
- Developing Applications on OpenFlow