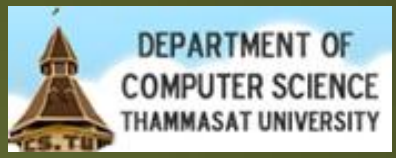


การใช้งานและติดตั้งระบบ OpenStack ซอฟต์แวร์สำหรับบริหารจัดการ Cloud Computing เบื้องต้น

Kasidit Chanchio
kasidit@cs.tu.ac.th
Thammasat University
Vasinee Siripoonya
Electronic Government Agency of Thailand
Phithak Thaenkaew
LSR, NECTEC


Outline

- Objectives
- Part I: OpenStack
  - Overview
  - How OpenStack components work
    - Keystone
    - Nova
    - Glance
- Part II: Demo
  - Use Cases

Objectives

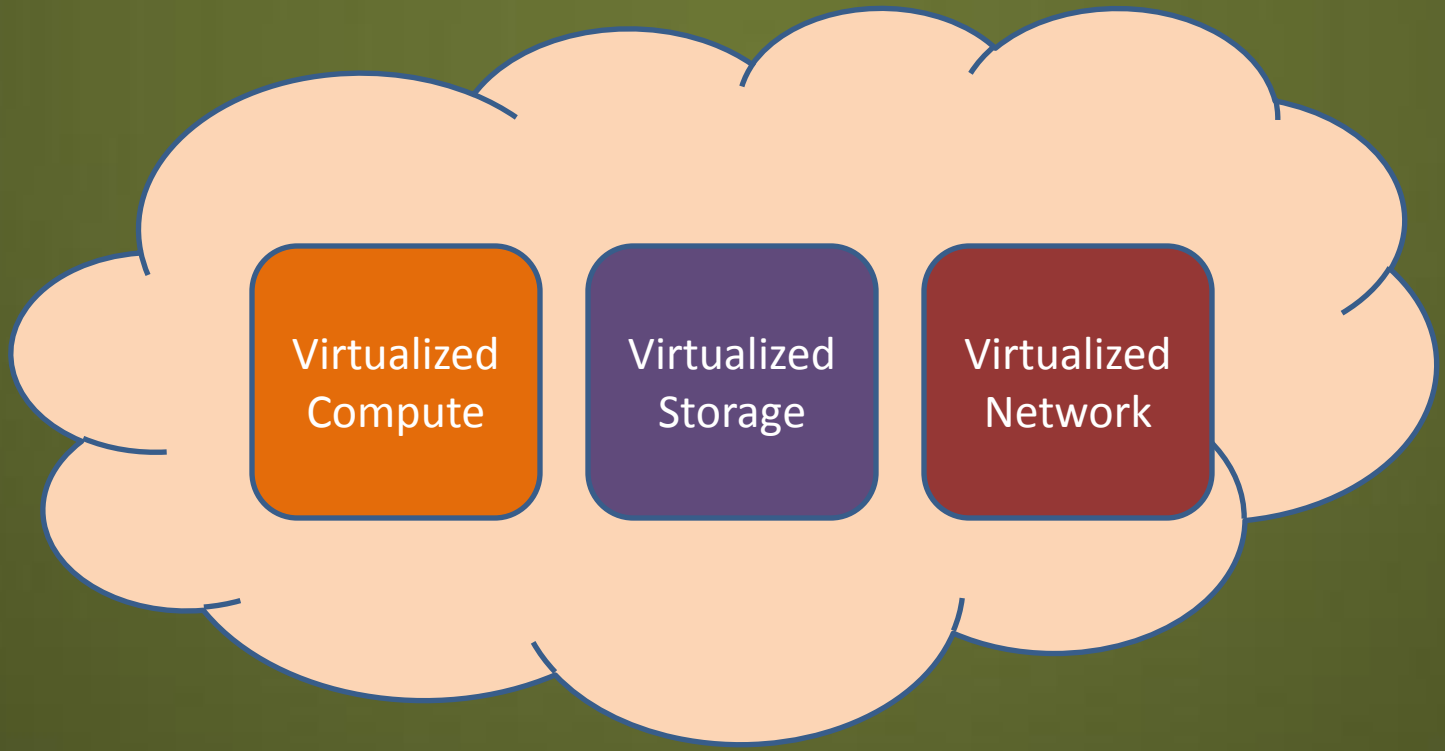
- ศึกษาเทคโนโลยี Cloud Computing เพื่อใช้เป็นพื้นฐานในการวิจัยและสร้างความรู้ใหม่
- ส่งเสริมการสร้างกลุ่มผู้ใช้งานและพัฒนาระบบ OpenStack หรือ OpenStack Community ในประเทศ
- สร้างความร่วมมือระหว่างองค์กรและผู้สนใจ
- ส่งเสริมการใช้งานและพัฒนาซอฟต์แวร์แบบ Open Source

Objectives to build our private cloud

- Create a Cloud Computing Platform to support research at Thammasat University and partner organizations
- Accumulate practical knowledge and experiences on Cloud deployment and operations
- Study OpenStack Cloud OS in order to integrate our advanced Fault Resilient and Cloud management mechanisms to it

Cloud

- Distributed Systems that provide **Services** to users “on-demand”
- Focus on “Infrastructure As A **Service**” Cloud



Public and Private Cloud

Public Cloud:
- Available over Internet
- "Pay-per-use" basis
- Resources are shared by users from anywhere

Private Cloud:
- Available over organization's IT infrastructure
- Pay by organization
- Resources are shared by users in same organization

Cloud Layers

Applications

Cloud OS

Virtualization/OS

Hardware/Storage/Network

Which Cloud OS should I use?

- Compatibility with your hypervisor/OS
  - Cloudstack comes from Citrix
  - OpenStack uses KVM by default. It has good support and documentation on Ubuntu
  - vCloud is definitely for vmware
- Maintainability
  - Provide means to fix the system when things go wrong
- Community Supports
- Etc.

Major OpenStack and CloudStack Supporters

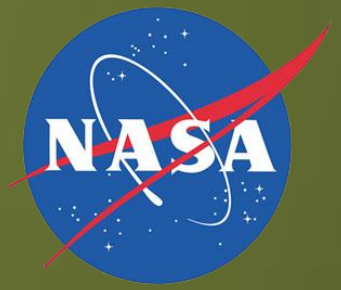
IT Vendors	OpenStack	CloudStack
Alcatel-Lucent		X
AMD	X	
Broadcom	X	
Brocade	X	X
Cisco	X	
Dell	X	
F5	X	
HP	X	
IBM	X	
Intel	X	X
Juniper	X	X
NEC	X	
NetApp	X	X
Red Hat	X	
Suse	X	
TrendMicro		X

Communications Service Providers	OpenStack	CloudStack
Akamai	X	
AT&T	X	
BT (British Telecom)		X
Deutsche Telekom	X	
Go Daddy		X
Internap	X	
KT (Korea Telecom)	X	X
NTT	X	X
Yahoo	X	

Source: DOMICITY LTD. – www.domicity.com

OpenStack® is an open and scalable cloud computing platform for building private and public clouds.

Invented by Rackspace and NASA.



The OpenStack project is provided under the Apache 2.0 license.

Participating Companies

Main Components

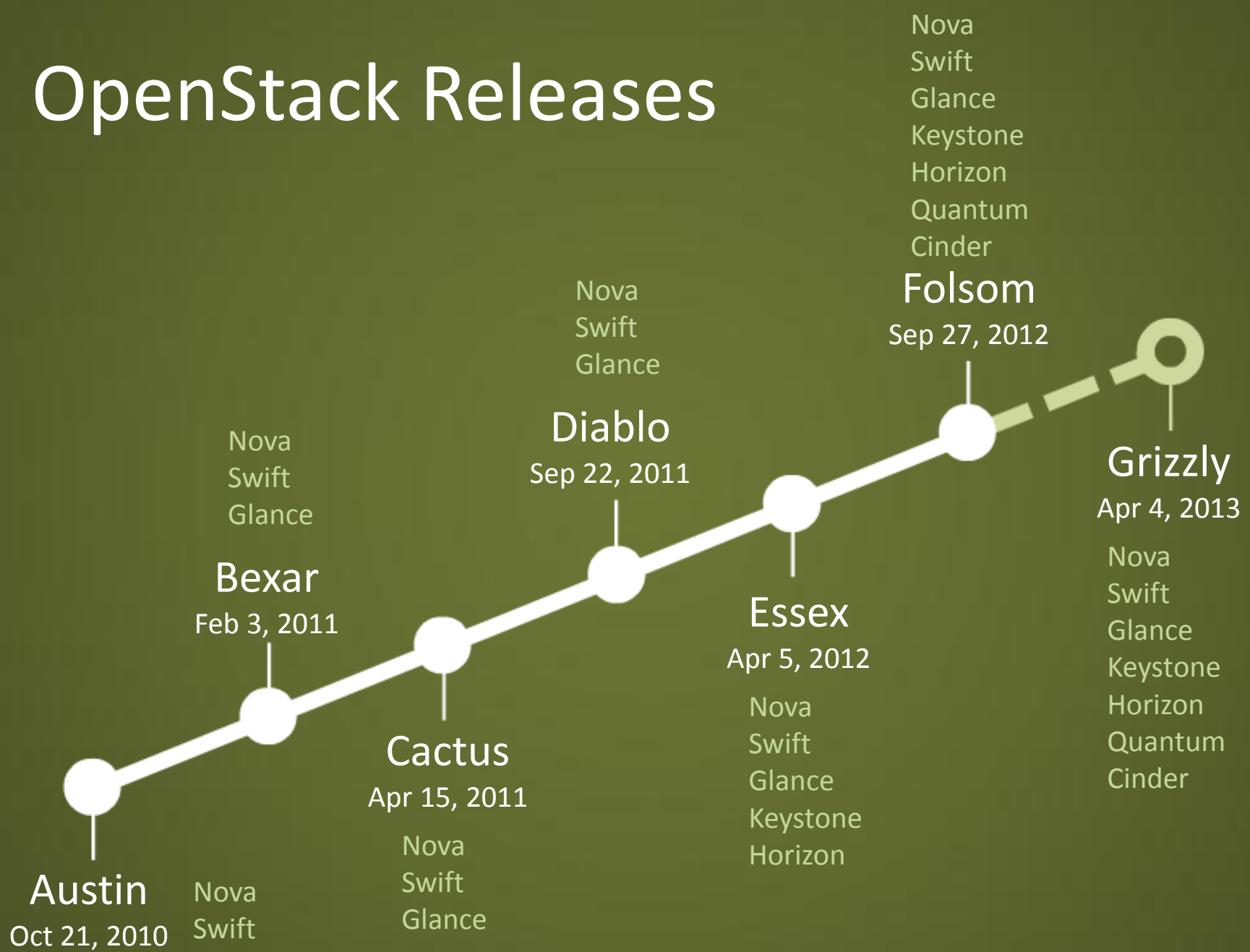
- OpenStack **Compute (Nova)**: Provision and manage large networks of virtual machines
- OpenStack **Object Storage (Swift)**: Create petabytes of secure, reliable storage using standard hardware
- OpenStack **Image Service (Glance)**: Catalog and manage massive libraries of server images

Main Components

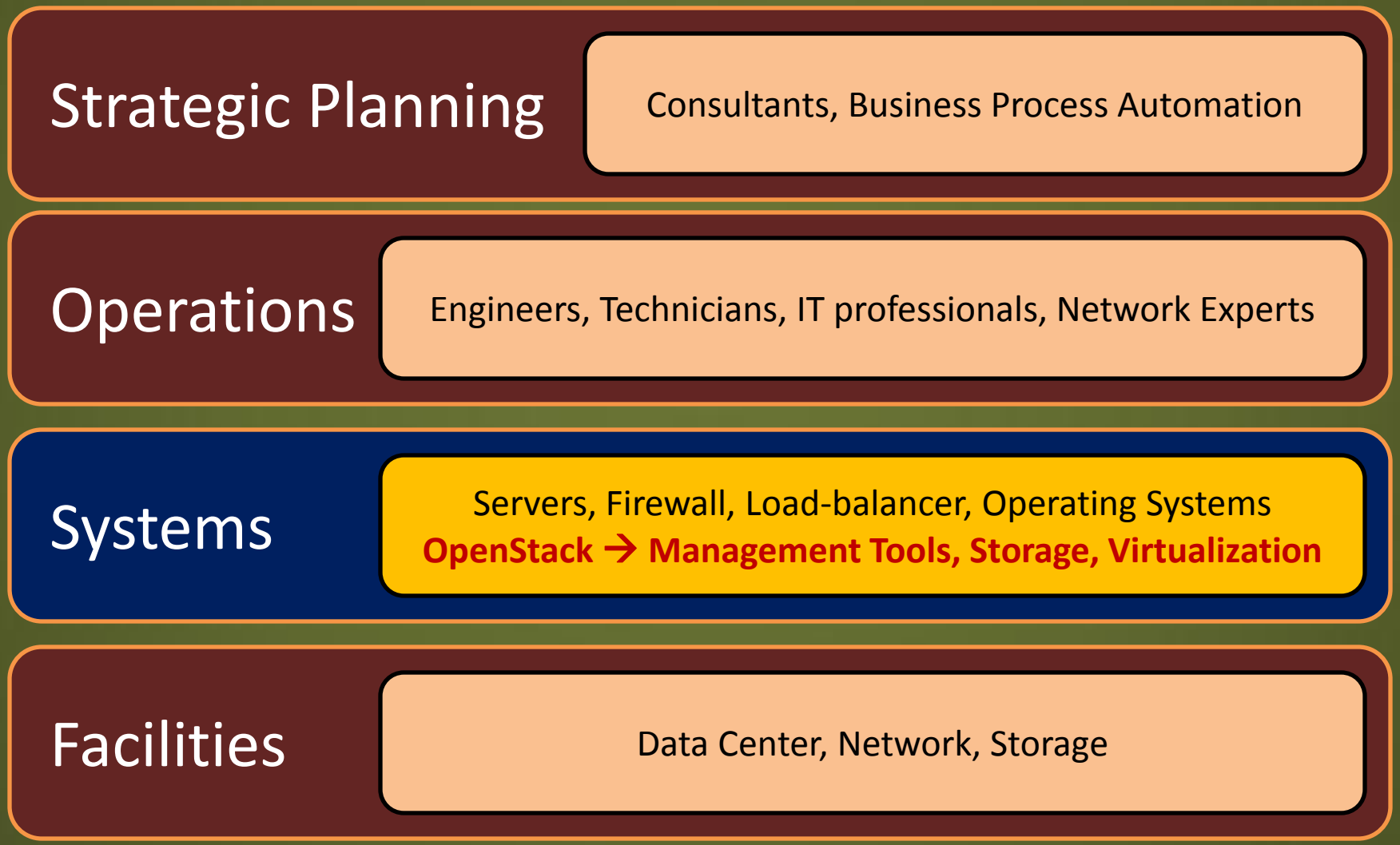
- OpenStack **Dashboard (Horizon)** : a modular web-based user interface for all the OpenStack services.

- OpenStack **Identity Service (Keystone)** : authentication and authorization for all the OpenStack services.

OpenStack Releases

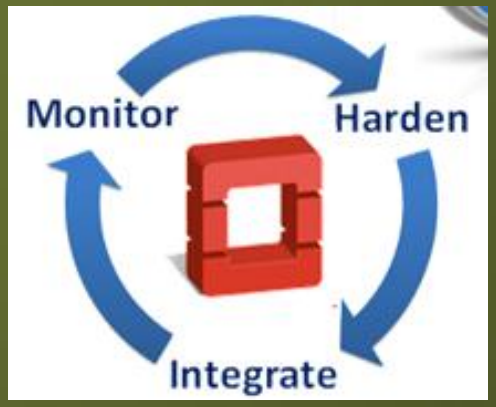


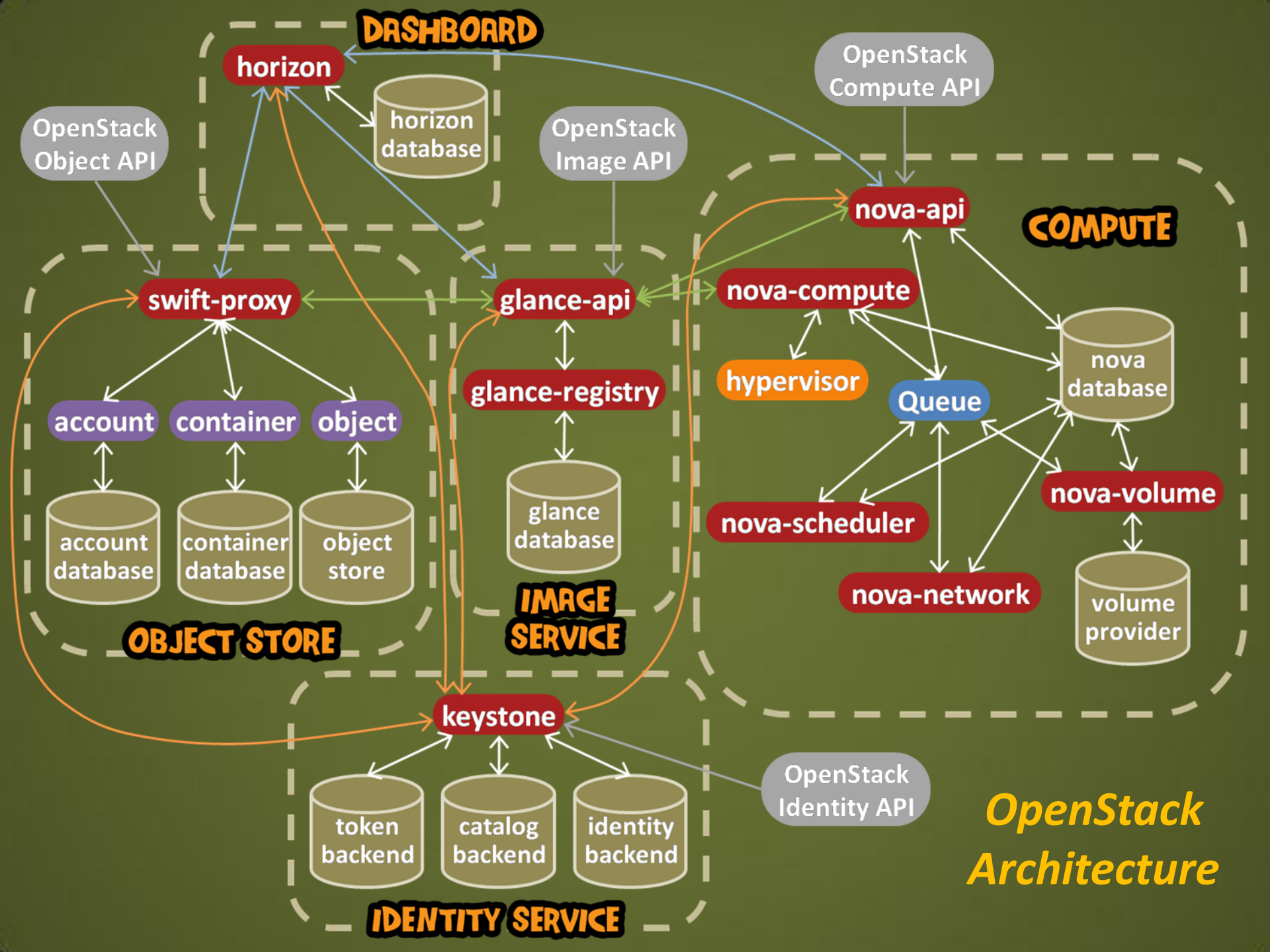
OpenStack isn't everything



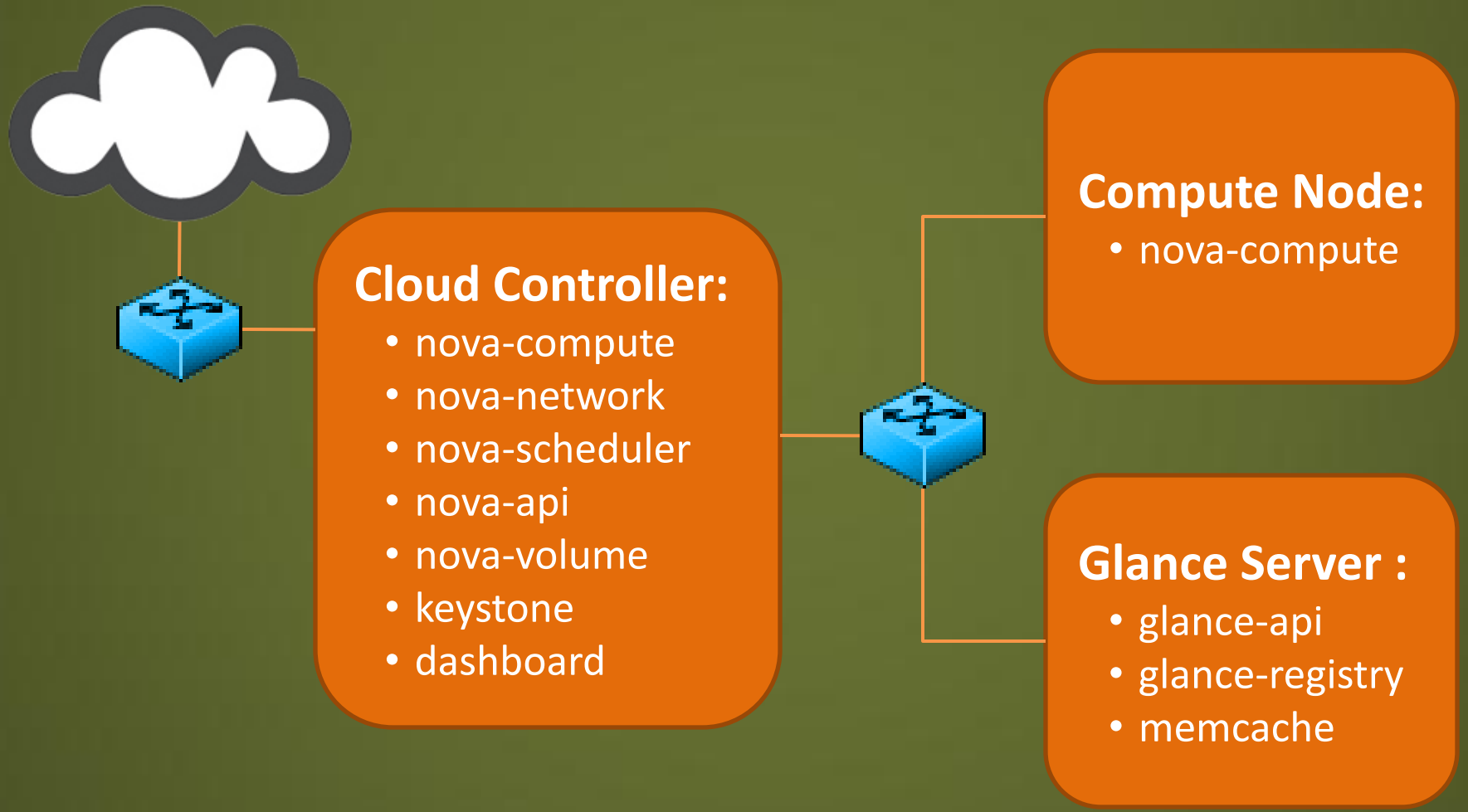
Hardening OpenStack Environments

- Restrict network and data access to least privilege
- Enable security features of underlying software
- Configure security features of underlying OS
- Harden the Hypervisor
- Use PKI for SSL
- Implement database security



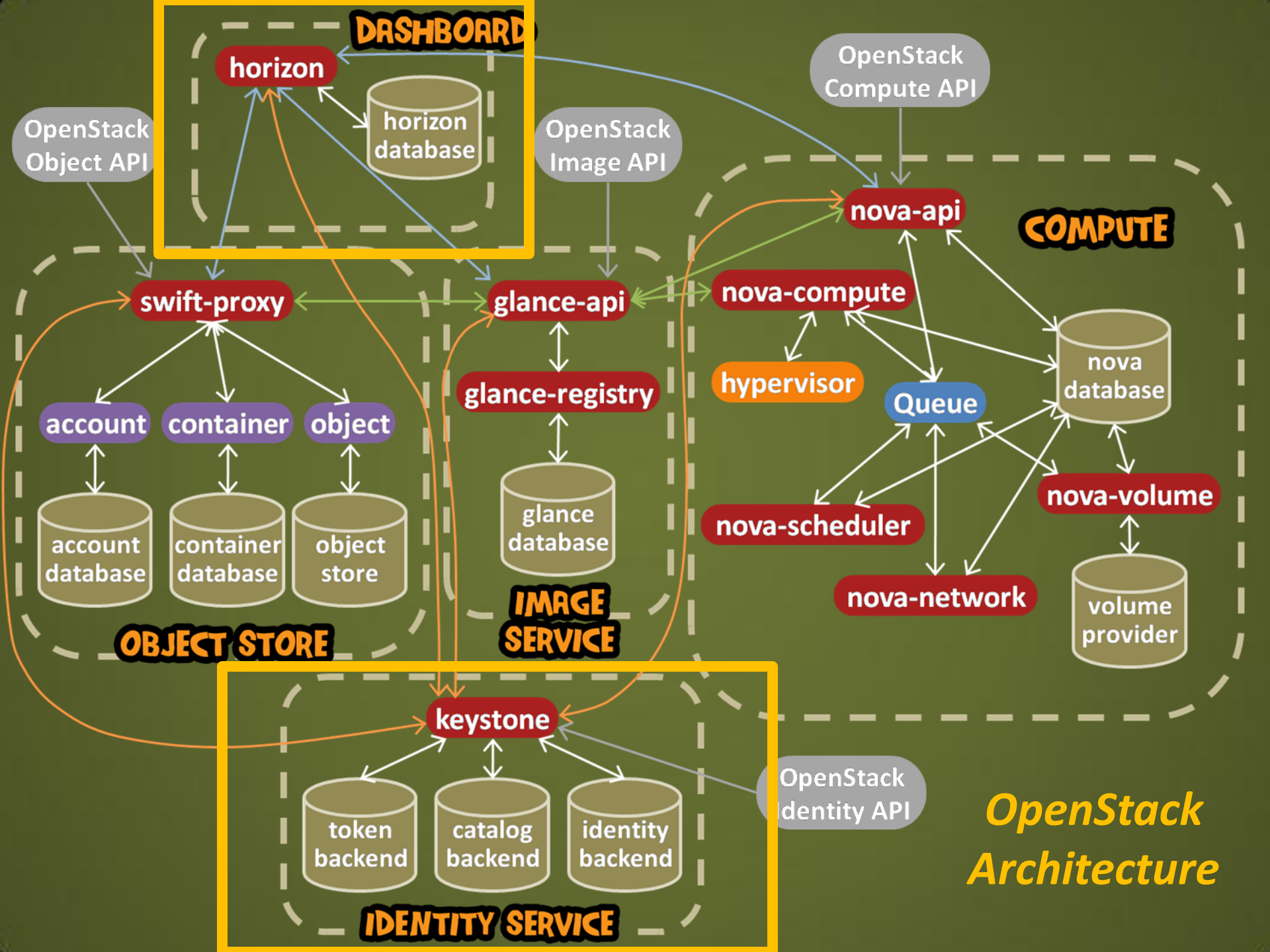

DASHBOARD
horizon
horizon database
OpenStack Object API
OpenStack Image API
OpenStack Compute API
nova-api
COMPUTE
swift-proxy
glance-api
nova-compute
hypervisor
Queue
nova database
glance-registry
account
container
object
account database
container database
object store
glance database
nova-volume
nova-scheduler
nova-network
volume provider
OBJECT STORE
IMAGE SERVICE
keystone
OpenStack Identity API
token backend
catalog backend
identity backend
IDENTITY SERVICE
OpenStack Architecture

SushiCloud's System Architecture



How OpenStack Components work

- Components in our focuses:
  - Keystone
  - Nova
  - Glance
- Networking Model
- Not currently cover
  - Swift


DASHBOARD
horizon
horizon database
OpenStack Object API
OpenStack Image API
OpenStack Compute API
nova-api
COMPUTE
swift-proxy
glance-api
nova-compute
hypervisor
Queue
nova database
glance-registry
account
container
object
account database
container database
object store
glance database
nova-volume
nova-scheduler
nova-network
volume provider
OBJECT STORE
IMAGE SERVICE
keystone
token backend
catalog backend
identity backend
OpenStack Identity API
IDENTITY SERVICE
OpenStack Architecture

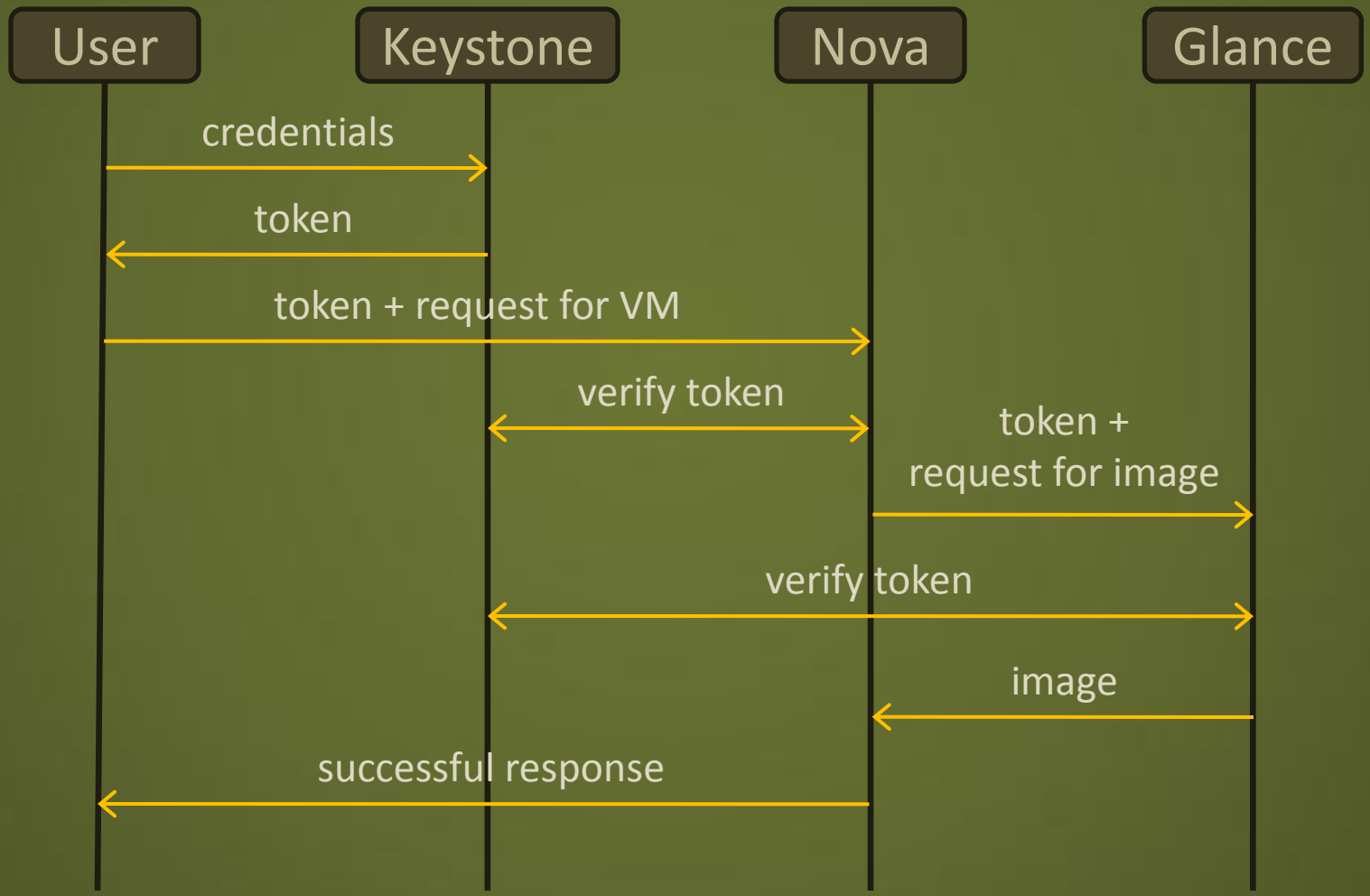
Keystone

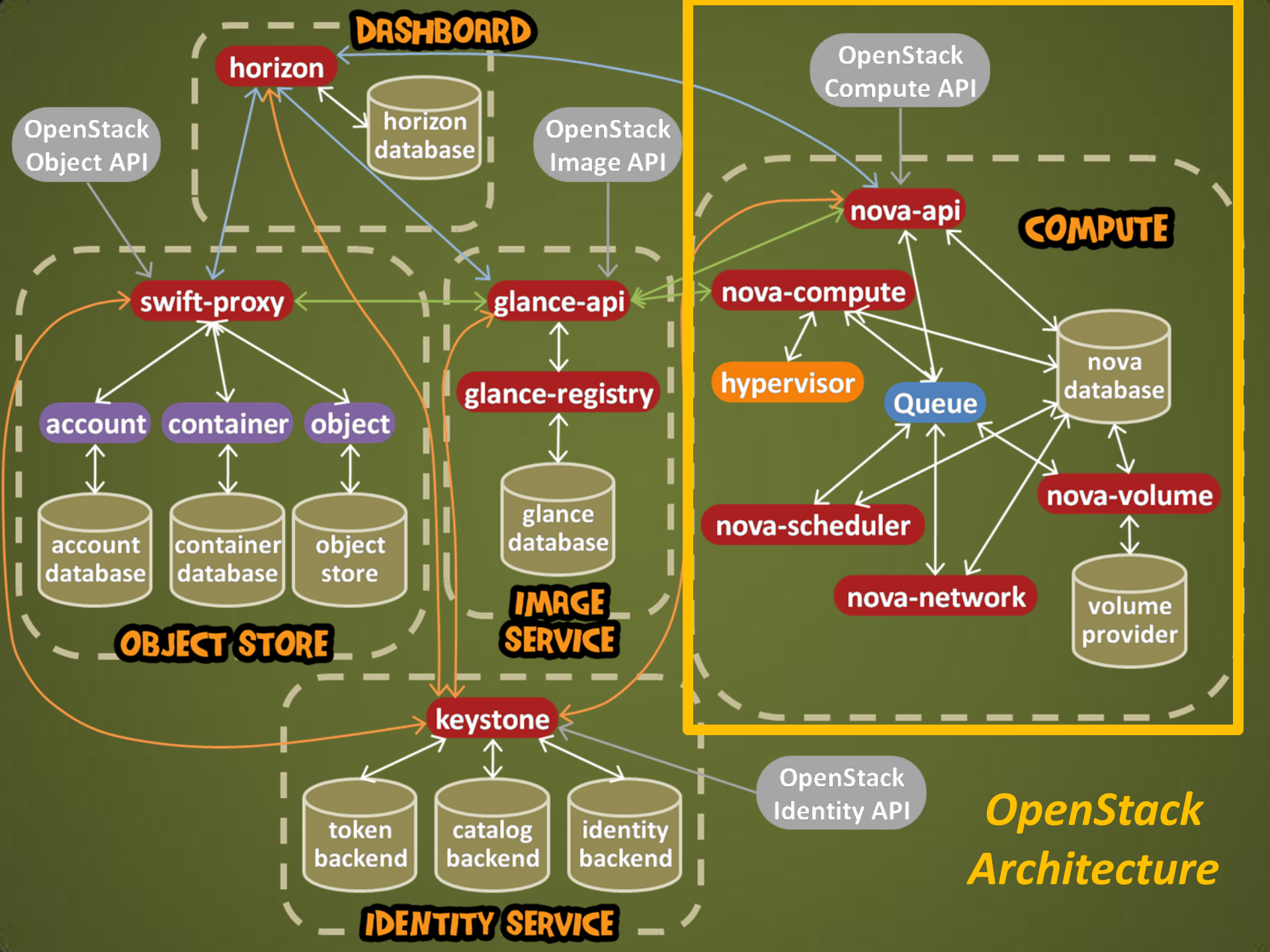
- A central authentication and authorization
- User represents someone or something that can gain access through Keystone. Users come with credentials that can be checked like passwords or API keys.
- Tenant represents what is called the project in Nova. Users are bound to a tenant by assigning them a role on that tenant.
- Role represents a number of privileges or rights a user has or actions they are allowed to perform.
- To access a service, we have to know its endpoint. So there are endpoint templates in Keystone that provide information about all existing endpoints of all existing services.

Keystone

- To access some service, users provide their credentials to Keystone and receive a token.
- If the user, for example, wants to spawn a new VM instance in Nova, one can find an URL to Nova in the list of endpoints provided by Keystone and send an appropriate request.
- After that, Nova verifies the validity of the token in Keystone and should create an instance from some image by the provided image ID and plug it into some network.
- All the way this token travels between services so that they can ask Keystone or each other for additional information or some actions.

Keystone Control Flow

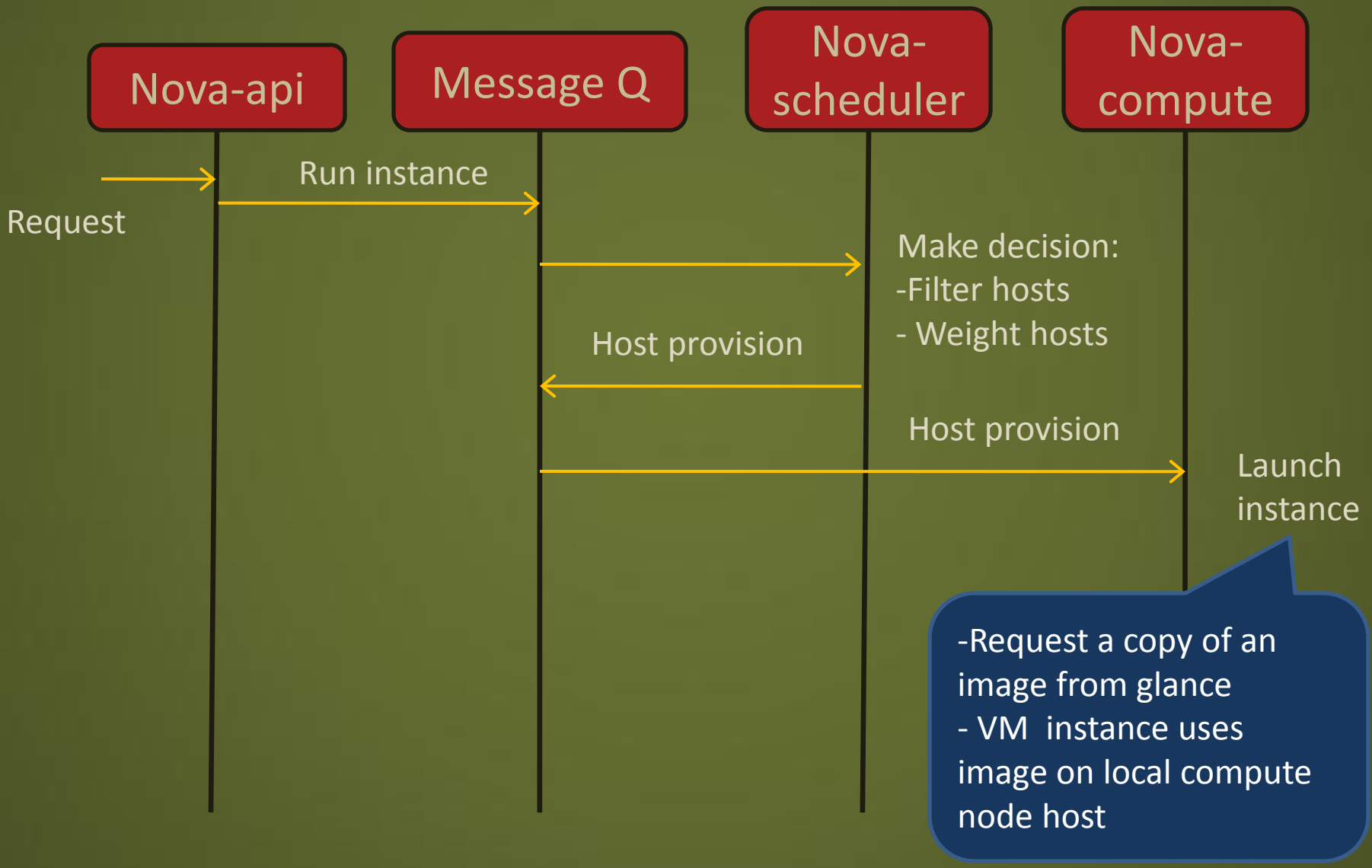



DASHBOARD
horizon
horizon database
OpenStack Object API
OpenStack Image API
OpenStack Compute API
COMPUTE
nova-api
swift-proxy
glance-api
nova-compute
hypervisor
Queue
nova database
nova-volume
nova-scheduler
nova-network
volume provider
glance-registry
account
container
object
account database
container database
object store
glance database
OBJECT STORE
IMAGE SERVICE
keystone
OpenStack Identity API
token backend
catalog backend
identity backend
IDENTITY SERVICE
OpenStack Architecture

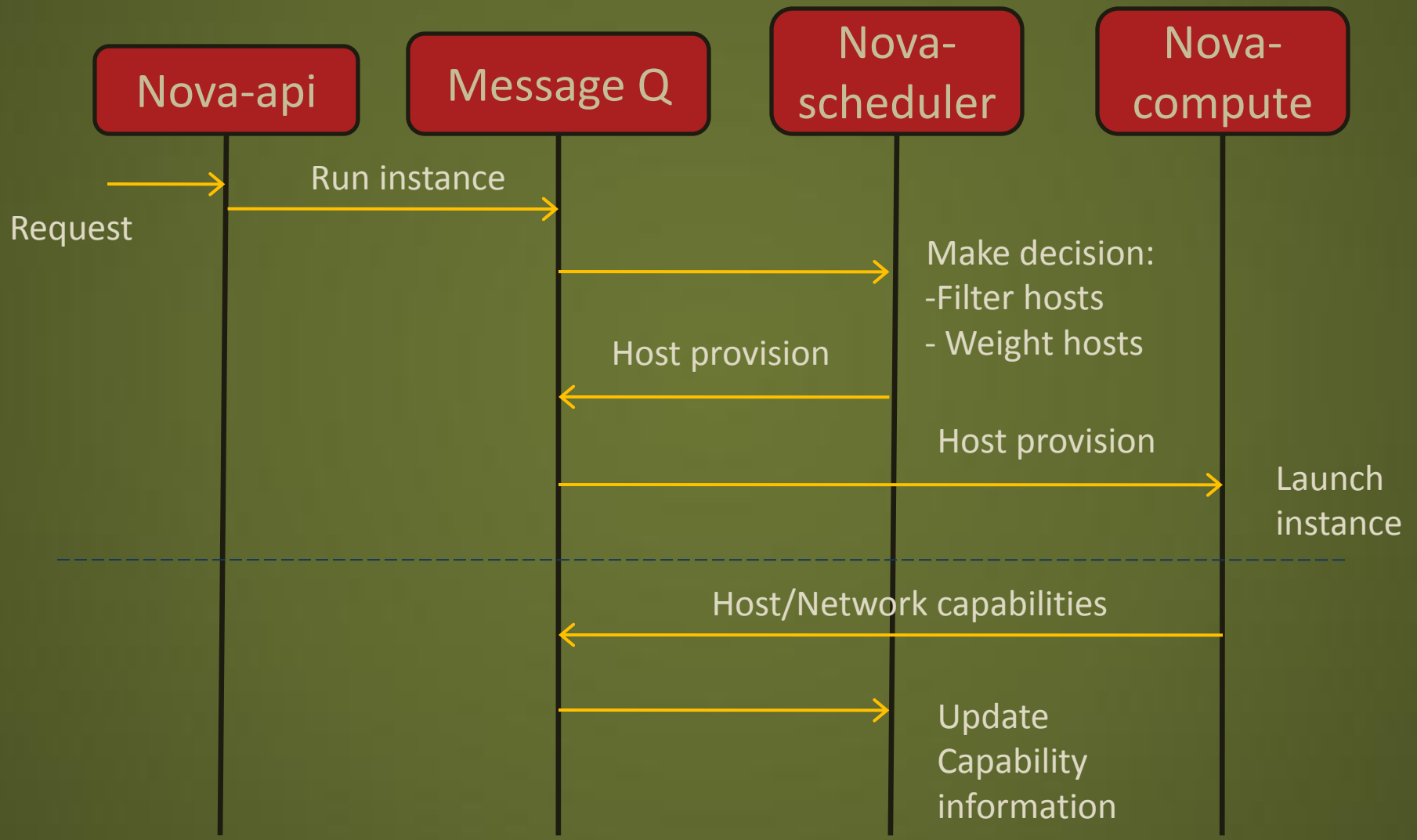
Nova

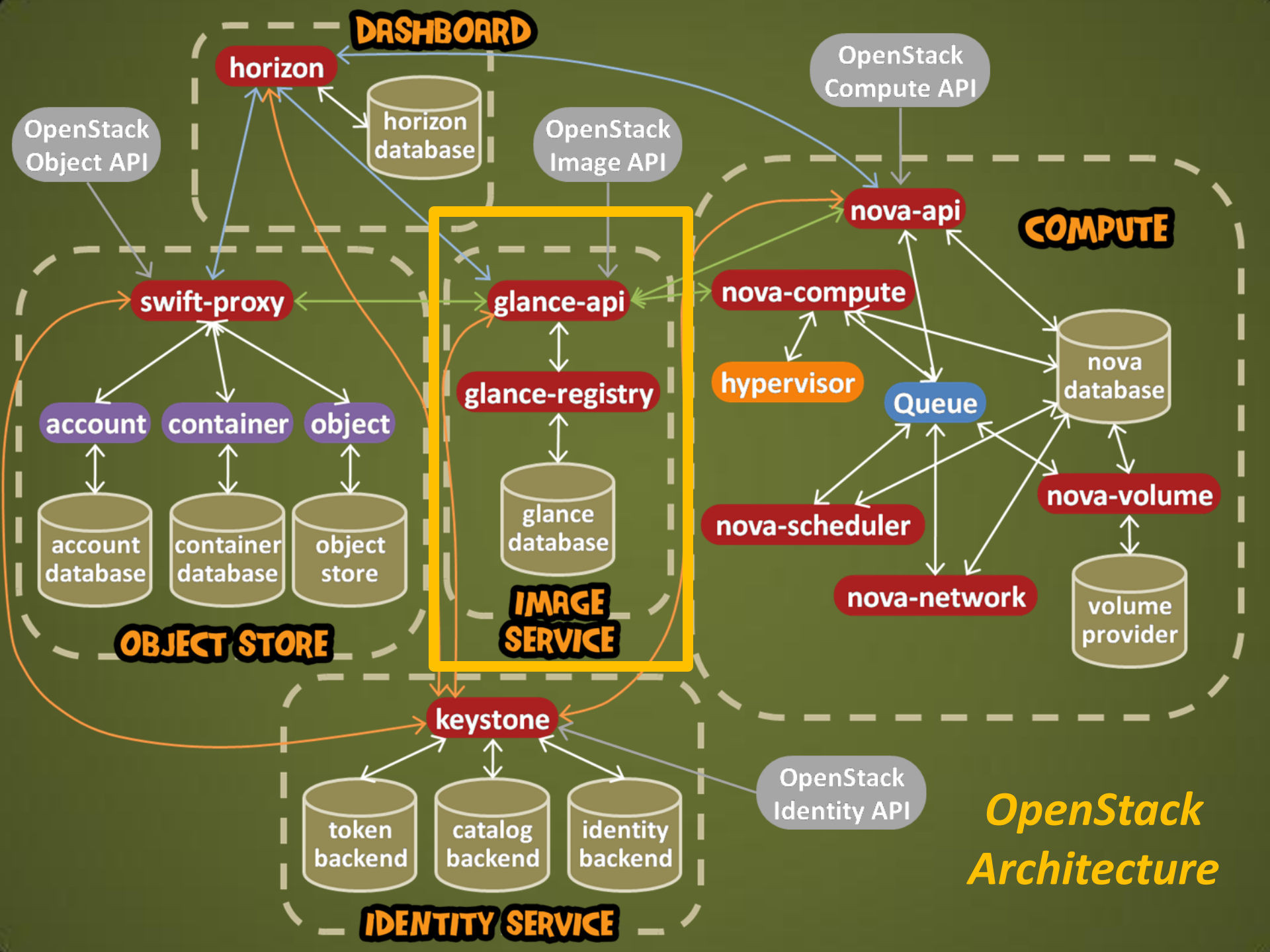
- Nova handles instances provisioning on compute resources.
- Nova-api initiates most activities
- Nova components communicate via queue and nova database
- Nova-scheduler decides where to launch instances
- Nova-compute launches instances
- Nova-compute periodically report host and network capabilities to Nova-scheduler

Nova Control Flow



Nova Control Flow

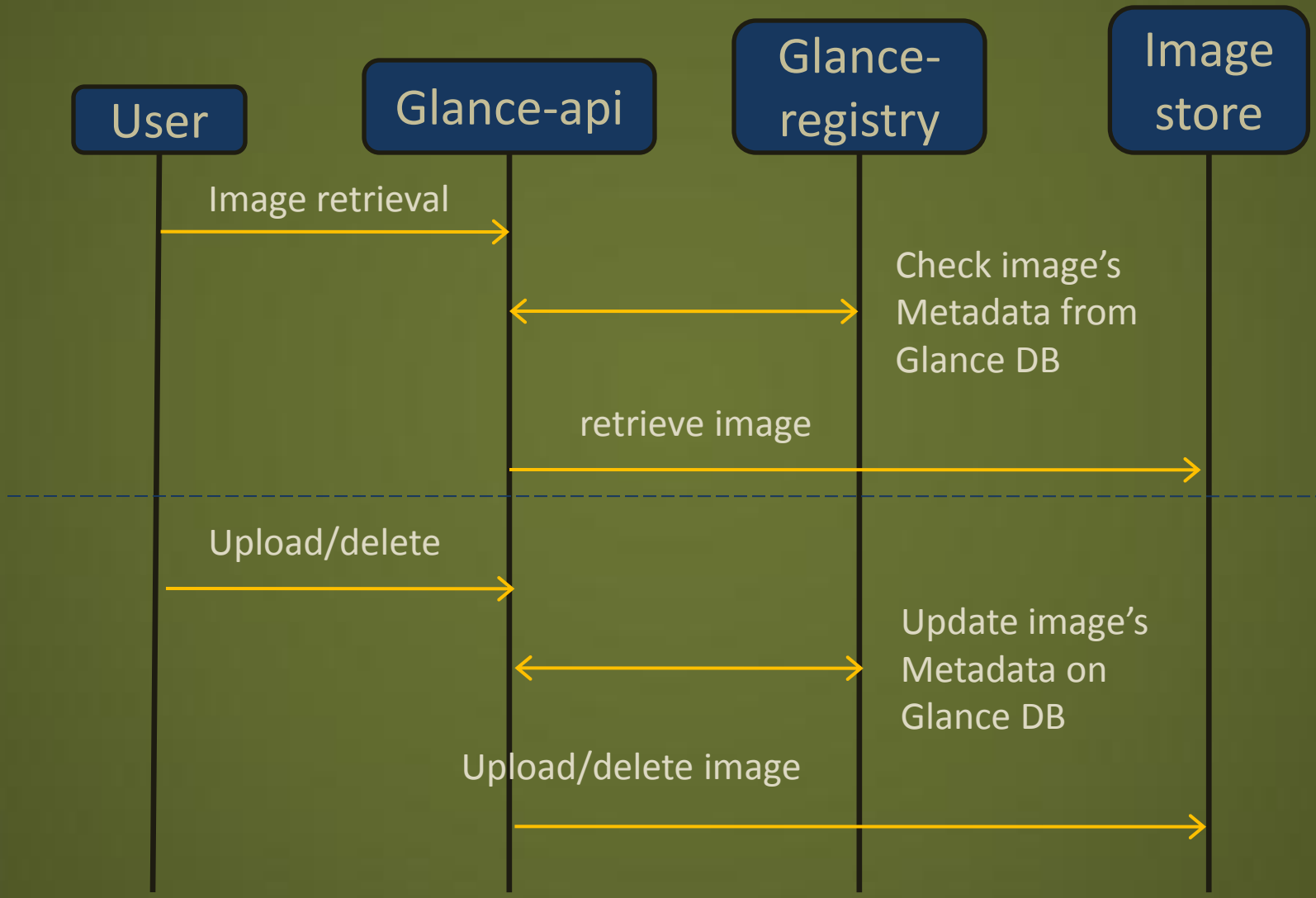



DASHBOARD
horizon
horizon database
OpenStack Object API
OpenStack Image API
OpenStack Compute API
COMPUTE
nova-api
swift-proxy
glance-api
nova-compute
glance-registry
hypervisor
Queue
nova database
account
container
object
account database
container database
object store
glance database
nova-volume
nova-scheduler
nova-network
volume provider
OBJECT STORE
IMAGE SERVICE
keystone
OpenStack Identity API
token backend
catalog backend
identity backend
IDENTITY SERVICE
OpenStack Architecture

Glance

- Glance manage all kinds of images to instantiate VM instances
- Glance-api takes image retrieval requests from nova-compute and pass them to glance-registry
  - OpenStack create **a new copy** of the image on a host where the VM instance runs
- Glance-registry check image metadata from database
- Glance stores Image data in its image store (S3, HTTP, Local, Swift)

Glance Control Flow



How OpenStack Components work

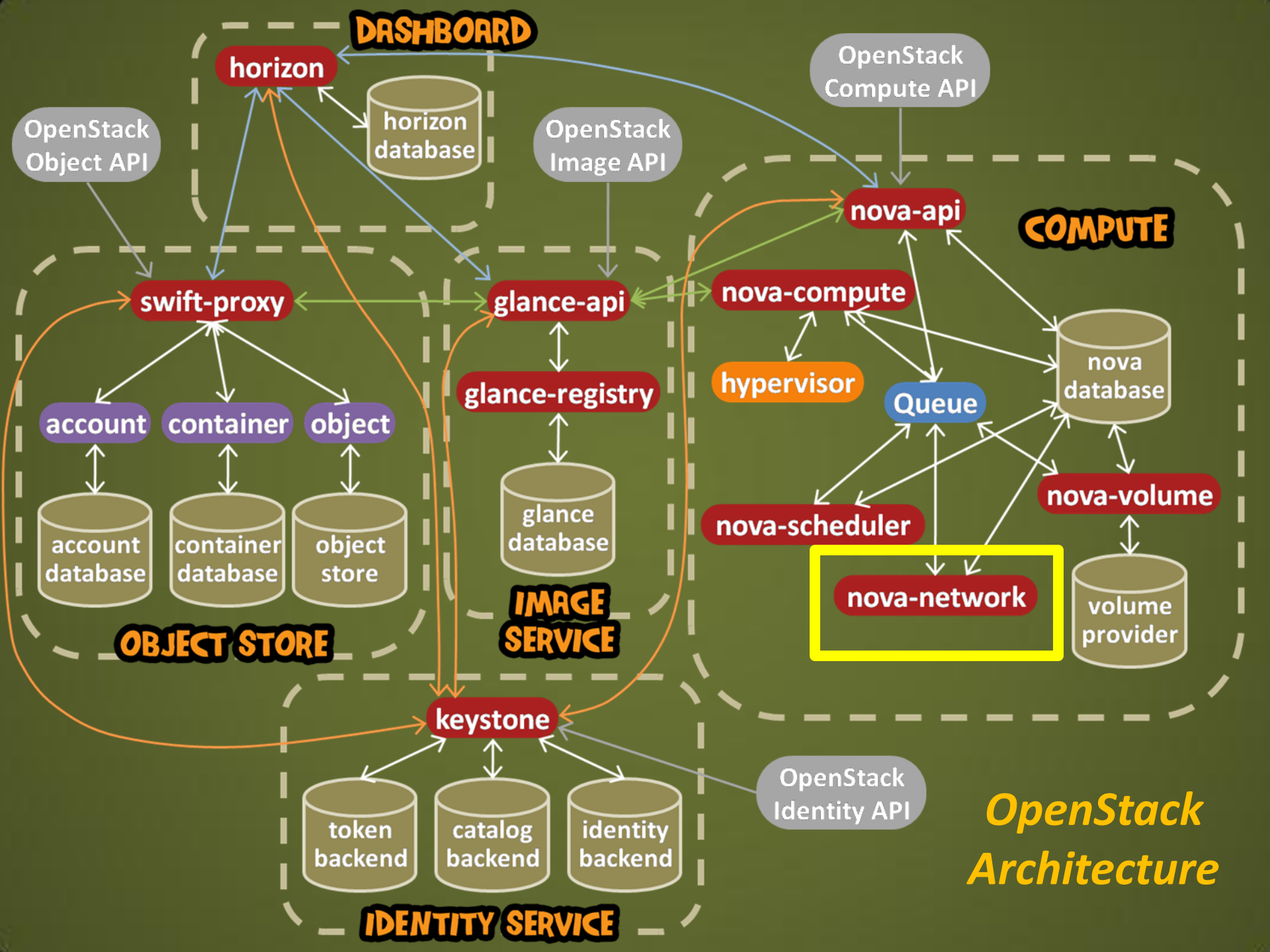
- Components in our focuses:
  - Keystone
  - Nova
  - Glance
- Networking Model
- Not currently cover
  - Swift

OpenStack Network Model

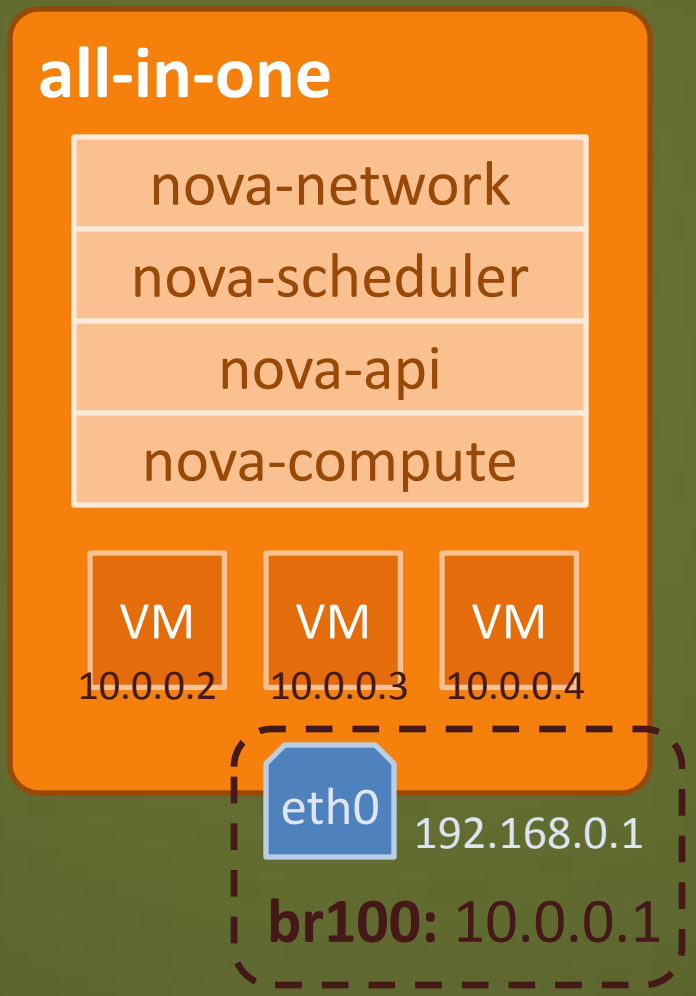
- Flat Network: A network administrator specifies a subnet from which all the virtual machines pulls IP addresses from a pool of available fixed addresses.
- Flat DHCP Network: The server that runs nova-network is a gateway to the compute nodes running virtual machines. Instances receive their fixed IPs by doing a dhcpdiscover. Like Flat Mode, all instances are attached to a single bridge on the compute node.
- VLAN Network: Compute creates a VLAN and bridge for each project. The project gets a range of private IPs that are only accessible from inside the VLAN. In this mode, each project gets its own VLAN, Linux networking bridge, and subnet.

IP address

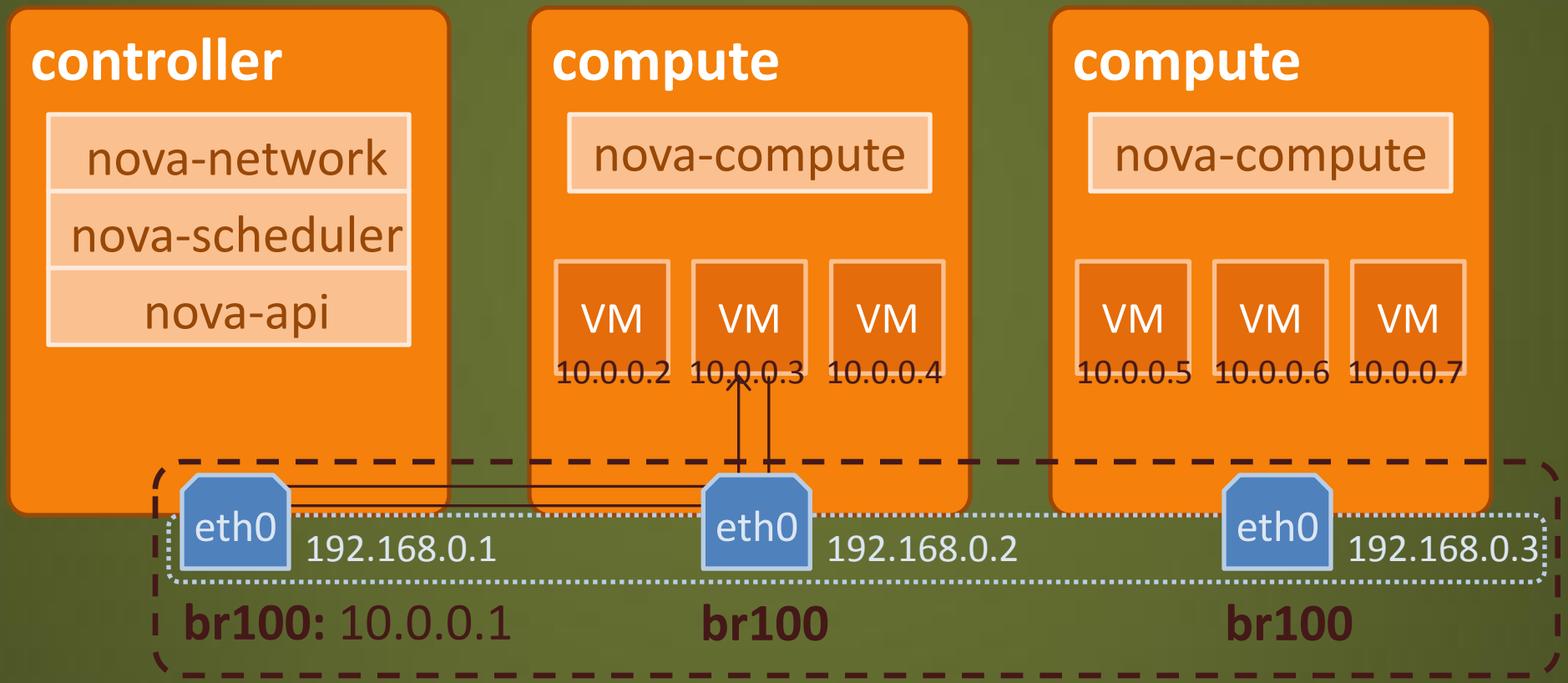
- Fixed IPs are IP addresses that are assigned to an instance on creation and stay the same until the instance is explicitly terminated.
- Floating IPs are addresses that can be dynamically associated with an instance. A floating IP address can be disassociated and associated with another instance at any time.


DASHBOARD
horizon
horizon database
OpenStack Object API
OpenStack Image API
OpenStack Compute API
nova-api
COMPUTE
swift-proxy
glance-api
nova-compute
hypervisor
Queue
nova database
glance-registry
account
container
object
account database
container database
object store
glance database
nova-volume
nova-scheduler
nova-network
volume provider
OBJECT STORE
IMAGE SERVICE
keystone
token backend
catalog backend
identity backend
OpenStack Identity API
IDENTITY SERVICE
OpenStack Architecture

Flat network, all-in-one server installation for development setup

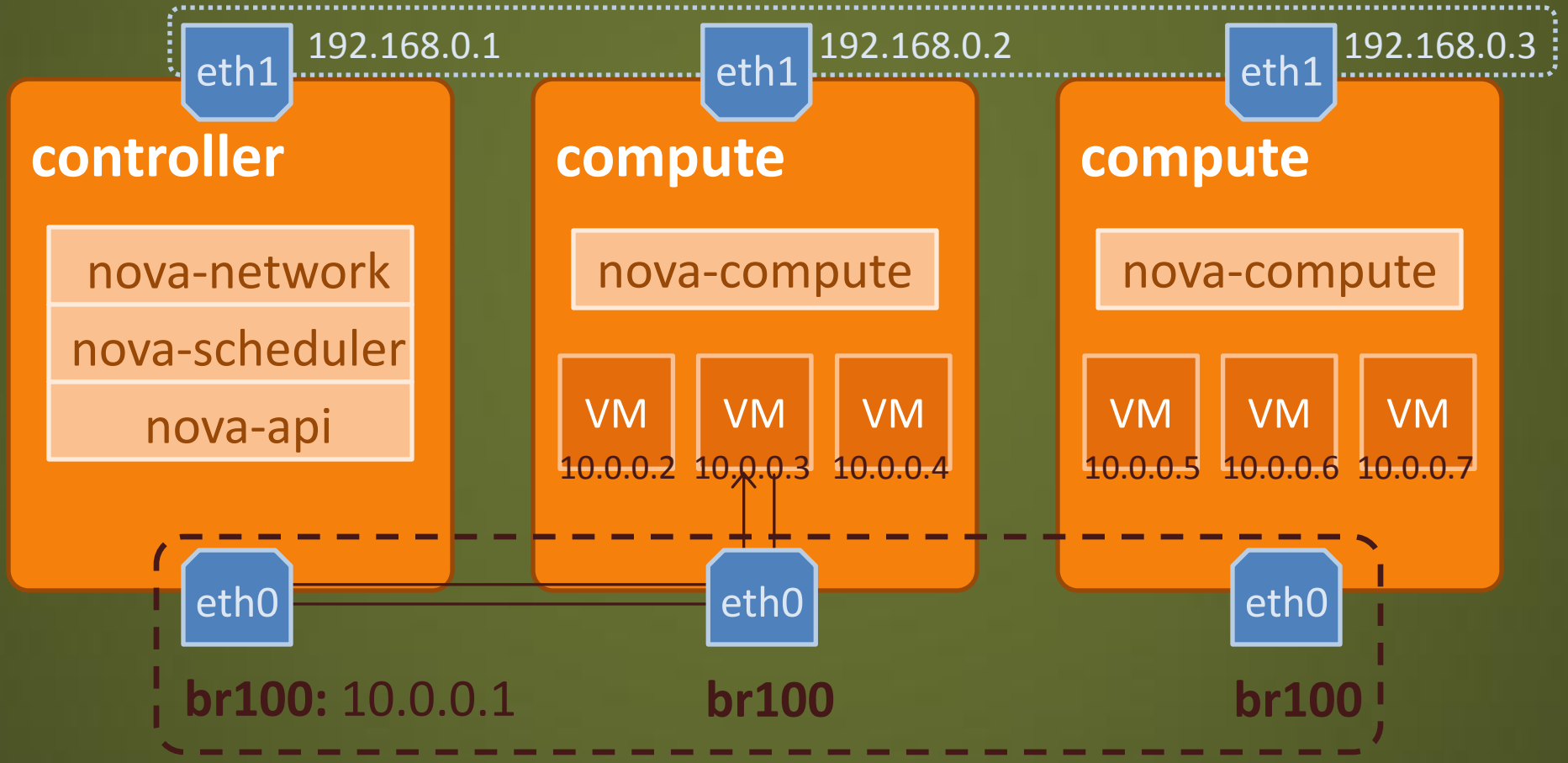


Flat network, multiple compute nodes with a single network adapter for smoke testing or a proof of concept

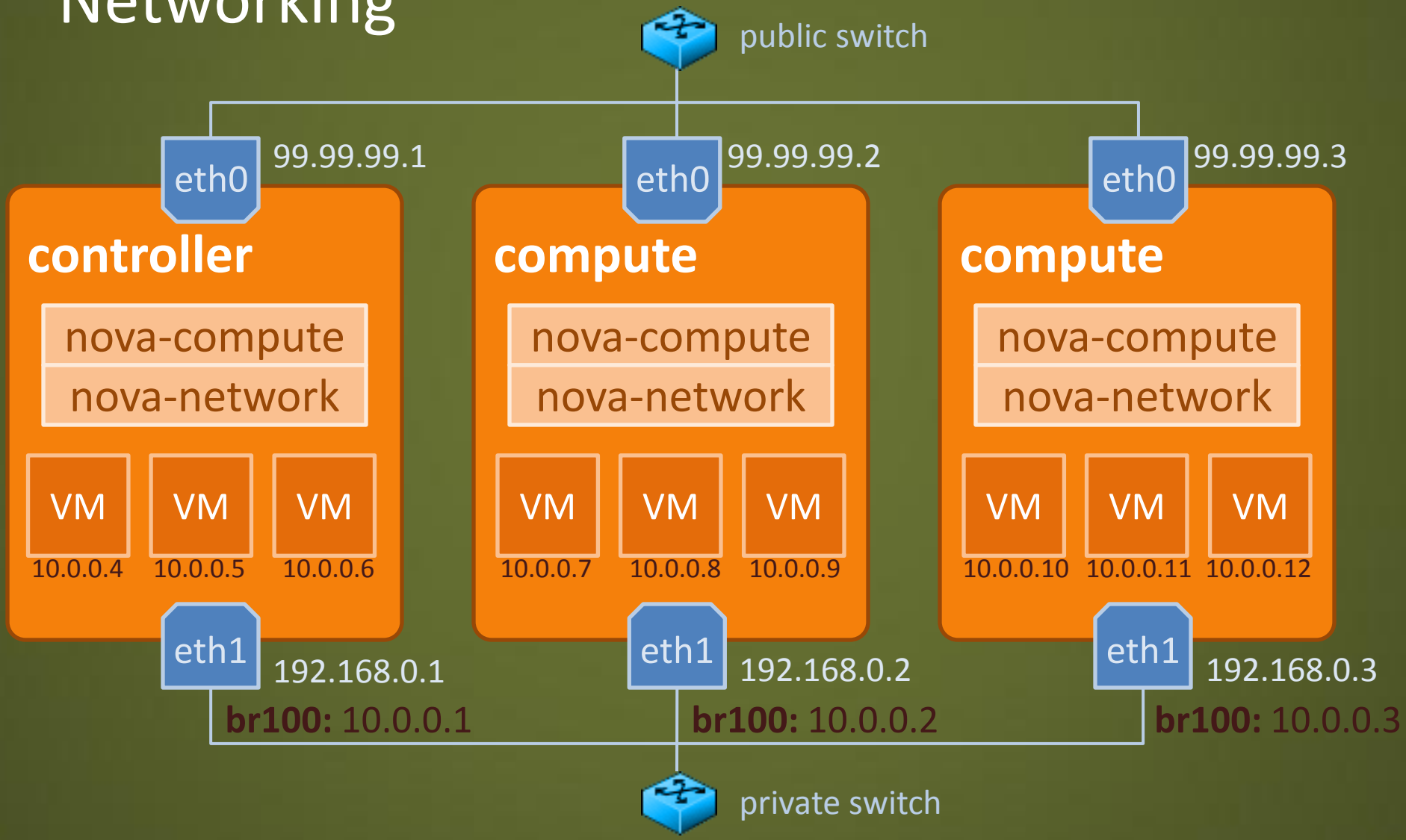


Note: OpenStack uses NAT to assign floating IP to VMs

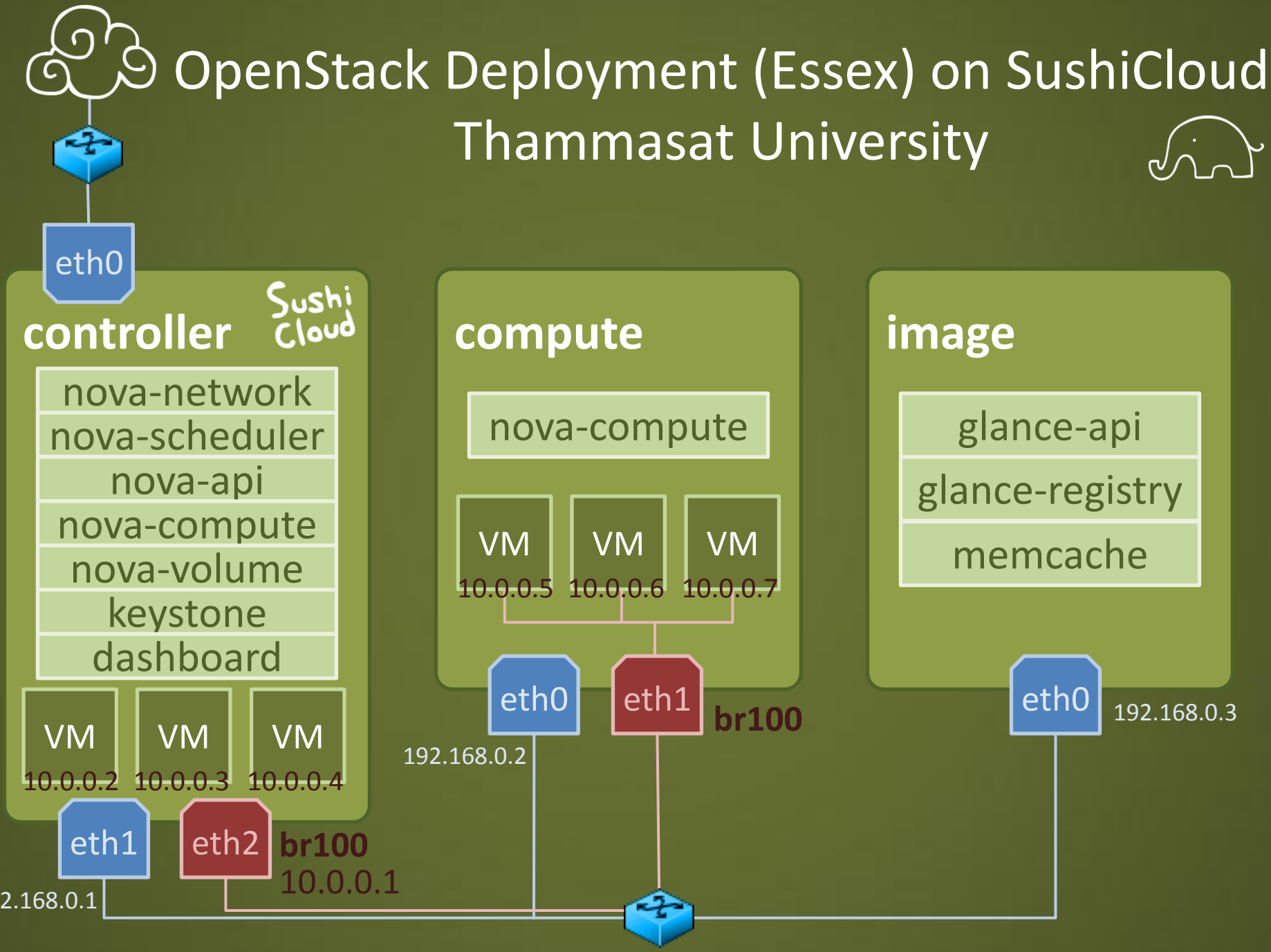
Flat network, multiple compute nodes with multiple network adapters for separate admin and data traffic



Flat DHCP network, multiple interfaces, multiple servers for High Availability Networking



OpenStack Deployment (Essex) on SushiCloud
Thammasat University



OpenStack Installation Instructions

- Scripted installation for proof-of-concept, learning, or development: DevStack (http://devstack.org/)
- Manual installation on Ubuntu, Debian, CentOS, Fedora or Red Hat Enterprise Linux 6 for deployment / production: OpenStack Manuals (http://docs.openstack.org/)
- Other Installation
  - ISO Distribution Installation: StackOps Distro - Community Edition / Enterprise Edition (http://www.stackops.com/)
  - Puppet Deployment Tool (dodai-deploy): OpenStack Manuals (http://docs.openstack.org/)

Hardware Recommendations

Recommended hardware configurations for a minimum production deployment for the cloud controller nodes

Server	Recommended Hardware	Notes
Cloud Controller node (runs network, volume, API, scheduler and image services)	Processor: 64-bit x86 Memory: 12 GB RAM Disk space: 30 GB (SATA or SAS or SSD) Volume storage: two disks with 2 TB (SATA) for volumes attached to the compute nodes Network: one 1 GB Network Interface Card (NIC)	32-bit processors will work for the cloud controller node. A quad core server with 12 GB RAM would be more than sufficient for a cloud controller node. Two NICS are recommended but not required.

http://docs.openstack.org/

Hardware Recommendations

Recommended hardware configurations for a minimum production deployment for the compute nodes

Server	Recommended Hardware	Notes
Compute nodes (runs virtual instances)	Processor: 64-bit x86 Memory: 32 GB RAM Disk space: 30 GB (SATA) Network: two 1 GB NICs	Note that you cannot run 64-bit VM instances on a 32-bit compute node. A 64-bit compute node can run either 32- or 64-bit VMs, however. With 2 GB RAM you can run one m1.small instance on a node or three m1.tiny instances without memory swapping, so 2 GB RAM would be a minimum for a test-environment compute node.

http://docs.openstack.org/

Compute and Image System Requirements

- **Operating System:** OpenStack currently has packages for the following distributions: CentOS, Debian, Fedora, RHEL, Debian, and Ubuntu.
- **Database:** For OpenStack Compute, you need access to either a PostgreSQL or MySQL database, or you can install it as part of the OpenStack Compute installation process.
- **Network Time Protocol:** You must install a time synchronization program such as NTP. For Compute, time synchronization keeps your cloud controller and compute nodes talking to the same time server to avoid problems scheduling VM launches on compute nodes.

Our Testbed Environment

- Cloud Controller node (runs network, volume, API, scheduler and compute)
- Compute node (runs virtual instances)
- Each node
  - Processor: 64-bit 12-cores 2.1 GHz Opteron
  - Memory: 48GB RAM
  - Disk space: 600GB
  - Volume storage: 300GB
  - Network: four 1Gbps NICs

Our Testbed Environment

- Glance node (runs image services)
- Processor: 64-bit two Quad core Xeon 5500 2.40Ghz
- Memory: 72GB RAM
- Disk space: 750GB
- Network: two 1Gbps NICs

Usage Scenario

- Once Upon a time, supposed you are a system admin of a small public cloud service provider company
- You want to create Cloud users and projects for a customer company
- Users access OpenStack to create and launch virtual machines by themselves

Using OpenStack Dashboard

- Add Users and Projects
- Create key-pair
- Launch Instance
- Configure Access
- Access Instance

Part II: Demo

SuhiCloud Testbed