

Creating and Accessing Instances: Compute and Image Services



Bernd Bausch

CLOUD TINKERER

<https://github.com/berndbausch/>



Overview



What's in a server:
Overview of server concepts

Launching a server

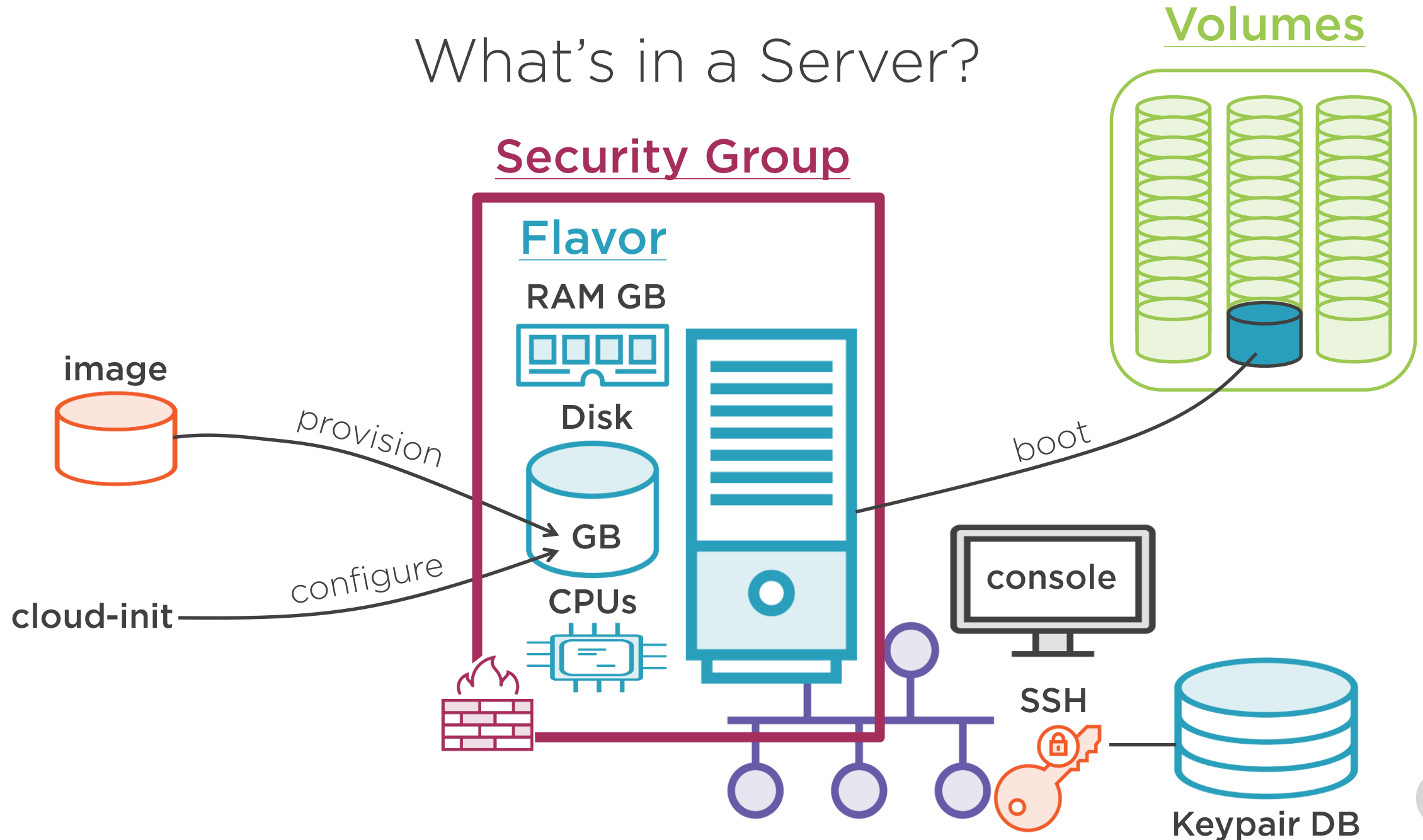
Console and network access

SSH keys

Personalizing the server:
Metadata and Userdata



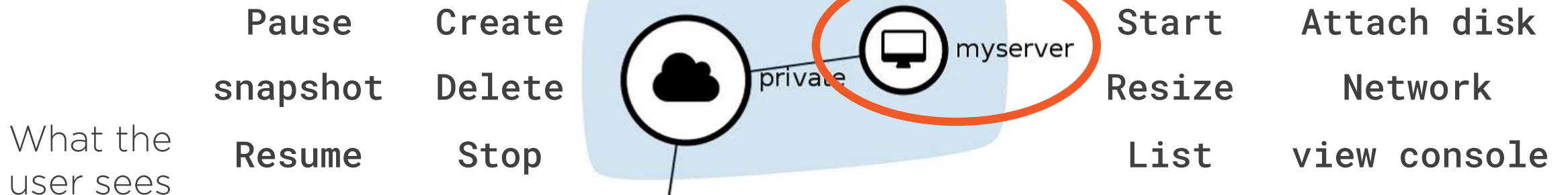
What's in a Server?



What's a Nova Server?

Also known as *instance*

It's an **abstraction** of virtual and physical server technologies



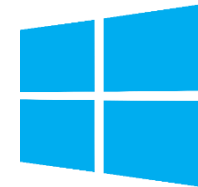
How it's implemented



vmware®



Linux Containers



Hyper-V



baremetal



Server Launch



What's in a server:
Overview of server concepts

Launching a server

Console and network access

SSH keys

Personalizing the server:
Metadata and Userdata

Other things you can do with a server



Launching a Server from the GUI

Launch Instance

Details *

Source *

Flavor *

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Instance source is the template used to create an instance. You can use a snapshot of an existing instance, an image, or a volume (if enabled). You can also choose to use persistent storage by creating a new volume.

Select Boot Source

Image

Image

Instance Snapshot

Volume

Volume Snapshot

Create New Volume

Yes

No

Delete Volume on Instance Delete

Yes

No

Allocated

Name	Updated	Size	Type	Visibility
Select an item from Available items below				

▼ Available 1

Select one

Q Click here for filters. X

Name	Updated	Size	Type	Visibility
...

X Cancel

< Back

Next >

Launch Instance



Launching a Server from the Command Line

Gathering Required Details

`$ openstack image list`

`$ glance image-list`

`$ openstack flavor list`

`$ nova flavor-list`

`$ openstack network list`

`$ neutron net-list`



Launching a Server from the Command Line

Reference = Name or ID

```
$ openstack server create
```

```
--image IMAGE_REFERENCE \
```

```
--flavor FLAVOR_REFERENCE \
```

```
--nic net-id=NETWORK_ID \
```

```
NAME_OF_INSTANCE
```

```
$ nova boot --image IMAGE_REFERENCE \
```

```
--flavor FLAVOR_REFERENCE --nic net-id=NETWORK_ID \
```

```
NAME_OF_INSTANCE
```



Universally Unique Identifier

What is an ID?

Almost everything has a UUID:

```
$ openstack server list
```

ID	Status	Networks
1797043c-1b42-4ec9-ae42-19874112c3be	titan1	ACTIVE
6dabb98d-0dca-4ffa-8e68-e3488d3c005d	titan1	ACTIVE

Contains time
and a node ID, e.g. MAC

```
$ openstack image list
```

ID	Name	Status
77a7b4bc-ae1a-4098-980c-dee8507471e3	cirros	active

Optional name
Doesn't have to be unique



What is an ID?

Almost everything has a UUID – with some exceptions:

```
$ openstack flavor list
```

ID	Name	RAM	Disk	Ephemeral	VCPUs	Is Public
1	m1.tiny	512	1	0	1	True
2	m1.small	2048	20	0	1	True
3	m1.medium	4096	40	0	2	True
4	m1.large	8192	80	0	4	True
5	m1.xlarge	16384	160	0	8	True



Listing, Starting, Stopping, Deleting Servers

```
$ openstack server list
```

```
$ nova list
```

```
$ openstack server show SERVER_REFERENCE
```

```
$ nova show SERVER_REFERENCE
```

```
$ openstack server stop SERVER_REFERENCE
```

```
$ nova stop SERVER_REFERENCE
```

```
$ openstack server start SERVER_REFERENCE
```

```
$ nova start SERVER_REFERENCE
```

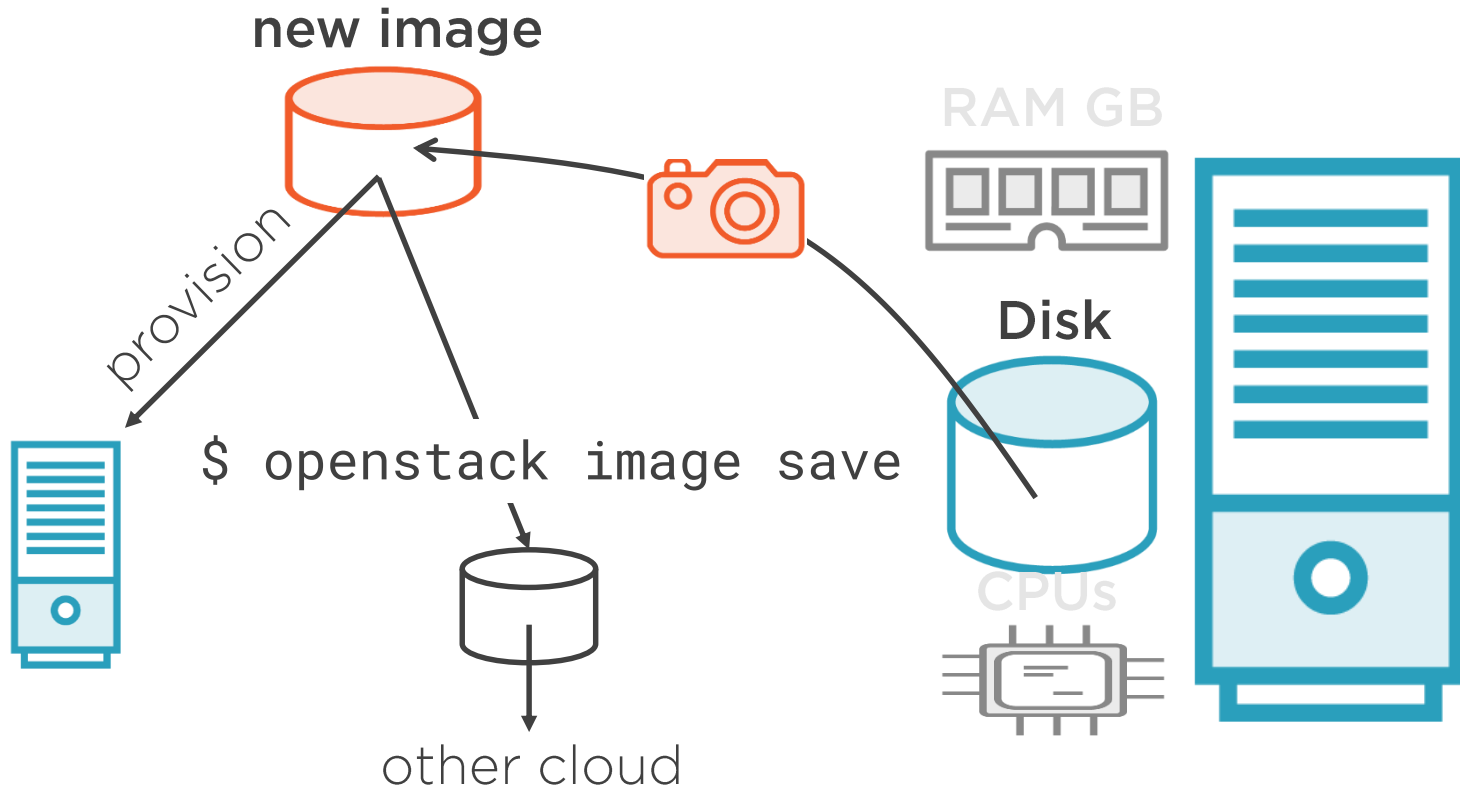
```
$ openstack server delete SERVER_REFERENCE
```

```
$ nova delete SERVER_REFERENCE
```

Data on server's disk
disappears



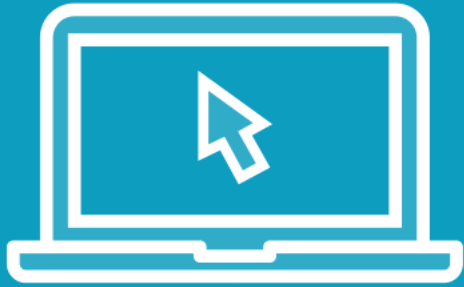
Server Snapshots



```
$ openstack server image create
$ nova image-create
```

<input type="checkbox"/>	double-ip	cirros	10.0.0.112	m1.tiny	demokey	Shutoff	nova	None	Shut Down	3 days, 19 hours	<div>Start Instance</div> <div>Create Snapshot </div>
--------------------------	-----------	--------	------------	---------	---------	---------	------	------	-----------	------------------	--

Demo



Launching, listing, snapshotting servers



Server Access



What's in a server:
Overview of server concepts

Launching a server

Console and network access

SSH keys

Personalizing the server:
Metadata and Userdata



```
$ openstack console log show
$ nova console-log
```

```
$ openstack console url show
$ nova get-vnc-console
```

```
Connected (unencrypted) to: QEMU (instance-00000006)
[ 3.834424] TCP cubic registered
[ 3.839423] NET: Registered protocol family 10
[ 3.885306] NET: Registered protocol family 17
[ 3.885663] Registering the dns_resolver key type
[ 3.944148] Freeing initrd memory: 3440k freed
[ 3.954458] registered taskstats version 1
[ 4.025625] usb 1-1: new full-speed USB device number 2 using uhci
[ 4.307439] Magic number: 9:165:809
[ 4.312680] rtc_cmos 00:01: setting system clock to 2017-07-05 00:4
1499215791)
[ 4.313171] powernow-k8: Processor cpuid 6d3 not supported
[ 4.316045] BIOS EDD facility v0.16 2004-Jun-25, 0 devices found
[ 4.316045] EDD information not available.
[ 4.366996] Freeing unused kernel memory: 924k freed
[ 4.430238] Write protecting the kernel read-only data: 12288k
[ 4.537218] Freeing unused kernel memory: 1600k freed
[ 4.626411] Freeing unused kernel memory: 1188k freed

further output written to /dev/ttyS0

login as 'cirros' user. default password: 'cubswin:). use 'sudo' for root.
myserver login:
login as 'cirros' user. default password: 'cubswin:). use 'sudo' for root.
myserver login: cirros
```

Instance Details: myserver

[Overview](#)[Log](#)[Console](#)[Action Log](#)

Log Length 35

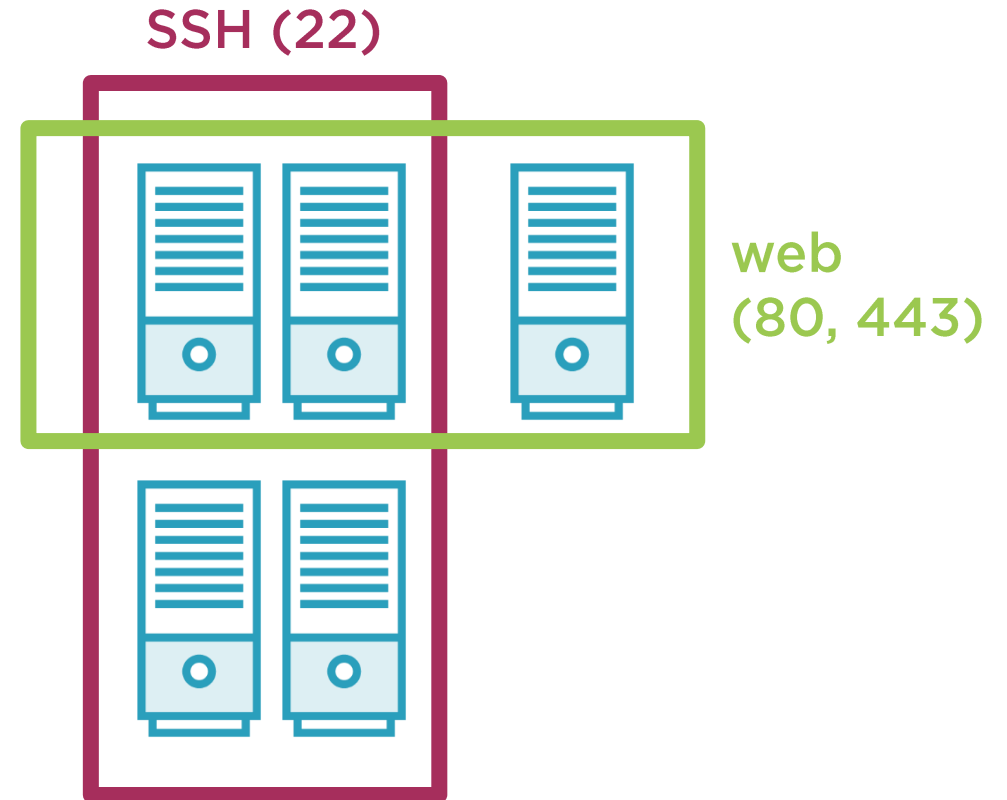
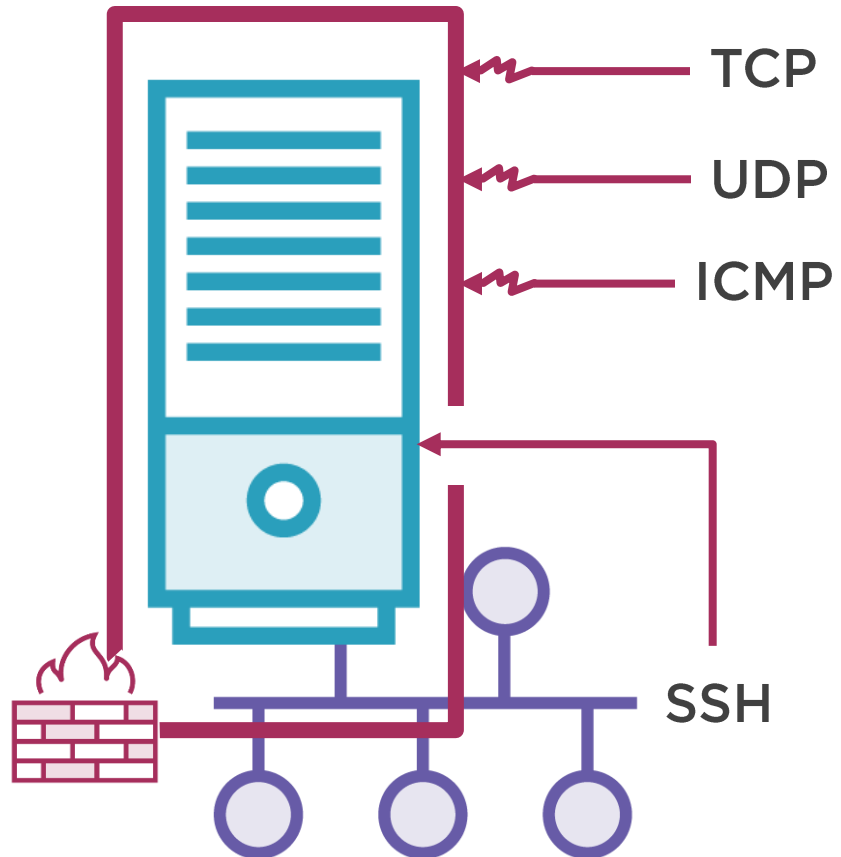
Instance Console Log

```
CPU(s): 1 @ 2491.890 MHz
Cores/Sockets/Threads: 1/1/1
Virt-type: AMD-V
RAM Size: 491MB
Disks:
NAME MAJ:MIN      SIZE LABEL      MOUNTPOINT
vda  253:0   1073741824
vda1 253:1   1061061120 cirros-rootfs /
=== sshd host keys ===
-----BEGIN SSH HOST KEY KEYS-----
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQnq0V5IFuvnvqIMUdF6Xx++WyNun/riXMrnKi3WwIk/IT1YddEPYs0
ssh-dss AAAAB3NzaC1kc3MAAACBAN0jltUwa3SKLS8gaEicbQxabcJ0Sw8FsmnwsLvBQYyr0icvLHEm+0xf4SePy0W
-----END SSH HOST KEY KEYS-----
=== network info ===
if-info: lo,up,127.0.0.1,8,::1
if-info: eth0,up,10.0.0.8,24,fe80::f816:3eff:fe21:b87f
ip-route:default via 10.0.0.1 dev eth0
ip-route:10.0.0.0/24 dev eth0  src 10.0.0.8
ip-route:169.254.169.254 via 10.0.0.1 dev eth0
```



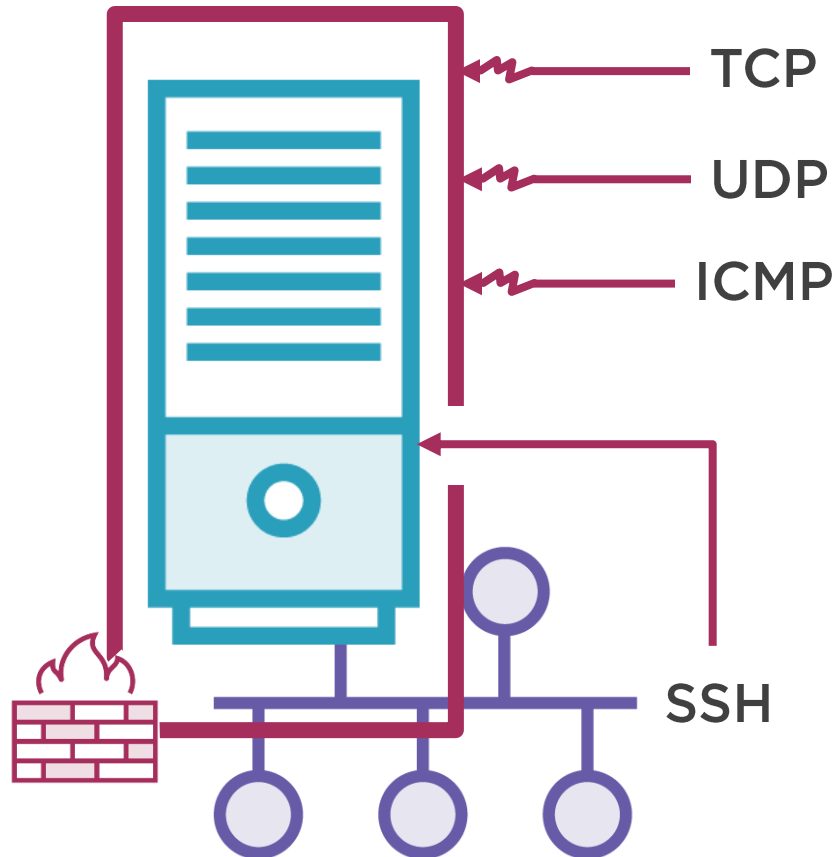
Security Groups

Security Group

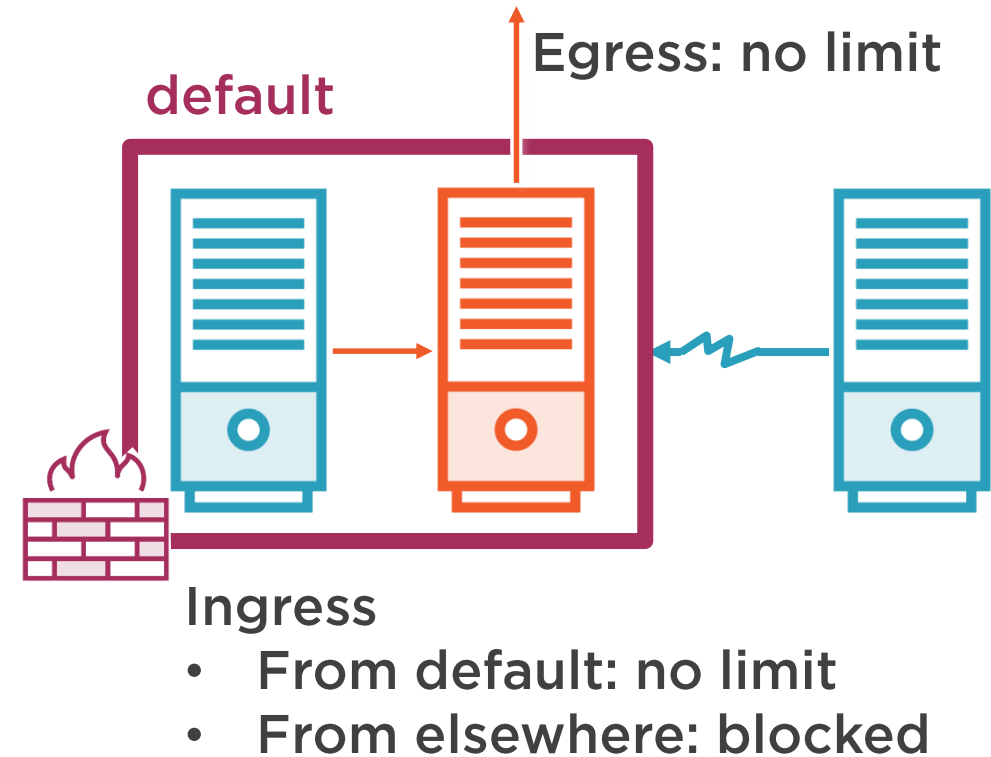


Security Groups

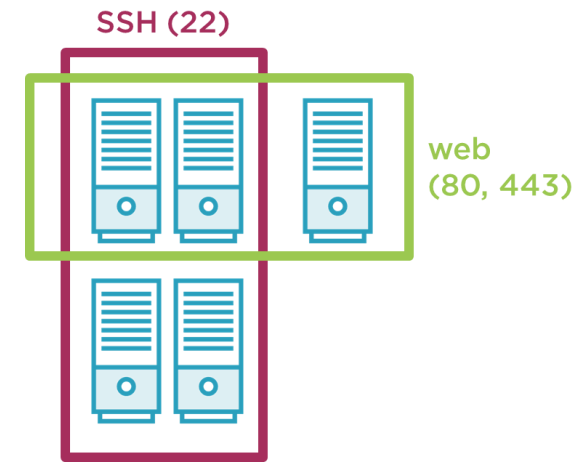
Security Group



Project's Default Security Group:



Security Groups in the GUI



Overview

Instances

Volumes

Images

Access & Security

Network >

Orchestration >

Object Store >

Access & Security

Security Groups

Key Pairs

Floating IPs

API Access

Filter



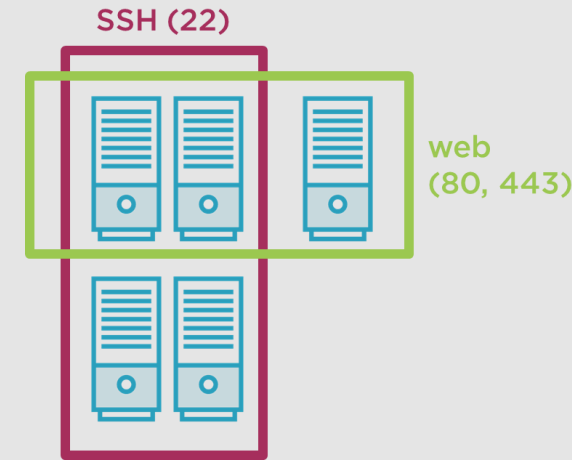
+ Create Security Group

Delete Security Group

<input type="checkbox"/>	Name	Description	Actions
<input type="checkbox"/>	default	Default security group	Manage Rules
<input type="checkbox"/>	ssh	Opens ports required for SSH and permits ICMP echo requests to reach the instance	Manage Rules
Displaying 2 items			



Security Group Commands



```
$ openstack security group create NAME
```

```
$ openstack security group rule create --protocol tcp --port 12345 NAME
```

Rule 1: "IPv4 egress allowed"

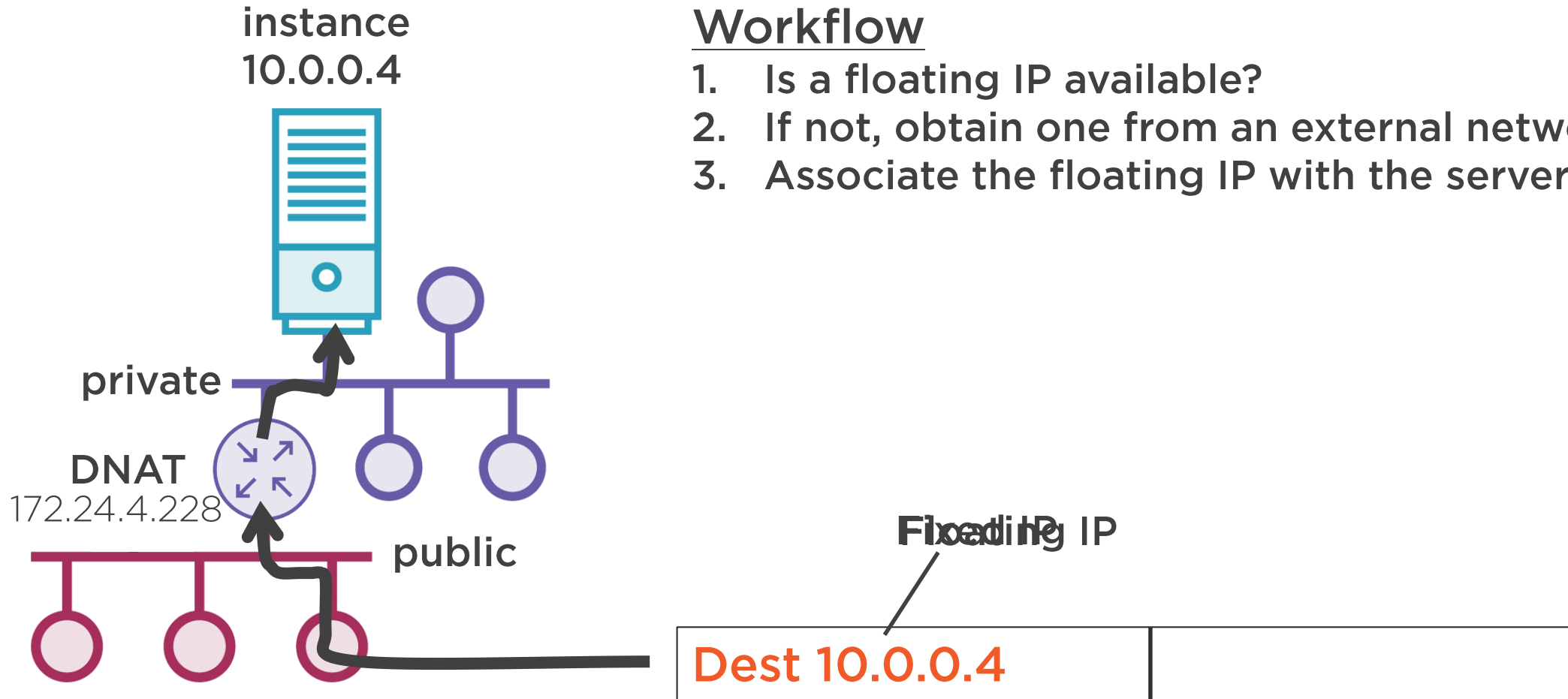
```
$ openstack server add security group
```

```
$ openstack security group rule list NAME
```

Floating IPs

Workflow

1. Is a floating IP available?
2. If not, obtain one from an external network
3. Associate the floating IP with the server



Floating IP in the GUI

1. No Floating IP available

2. Create one

3. Associate it with a server

The screenshot displays the OpenStack GUI's 'Access & Security' section. The 'Floating IPs' tab is active, showing a table of floating IP addresses. A green box highlights the 'Allocate IP To Project' button, and another green box highlights the 'Associate' button in the actions column. Arrows from the numbered text blocks point to these elements.

IP Address	Mapped Fixed IP Address	Pool	Status	Actions
<input type="checkbox"/> 172.24.4.230	webserver1 10.10.10.6	public	Active	Disassociate
<input type="checkbox"/> 172.24.4.237	-	public	Down	Associate
<input type="checkbox"/> 172.24.4.229	webserver2 10.10.10.5	public	Active	Disassociate

Displaying 3 items

Floating IP Commands

```
$ openstack floating ip list
```

```
$ openstack floating ip create EXTERNAL_NETWORK
```

```
$ openstack server add floating ip SERVER IP_ADDRESS
```

```
$ openstack server remove floating ip SERVER IP_ADDRESS
```

```
$ openstack floating ip delete IP_ADDRESS
```

Be a good citizen or
reduce the bill



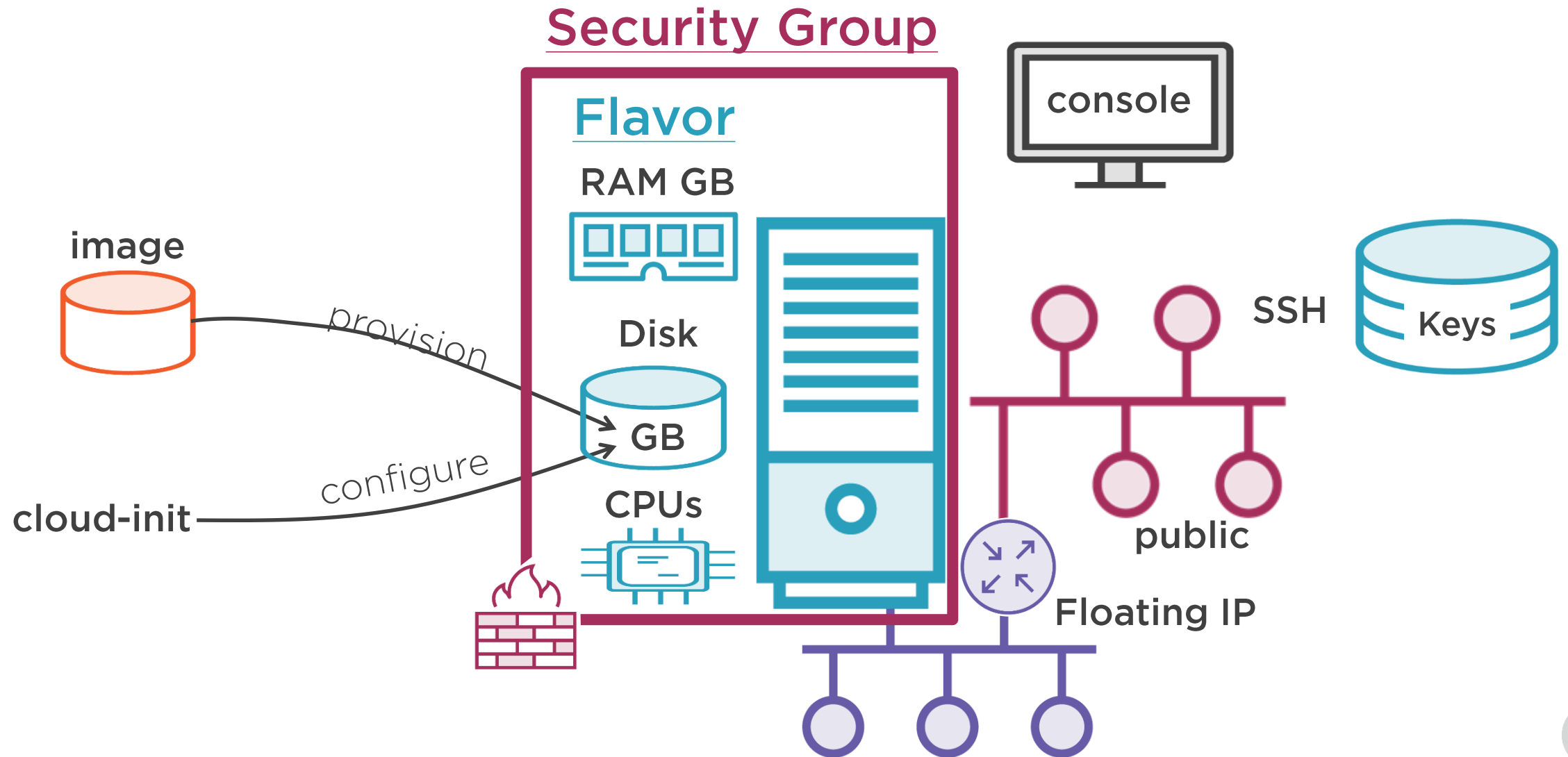
Demo



Console and network access



What Have We Done so Far?



No Password



What's in a server:
Overview of server concepts

Launching a server

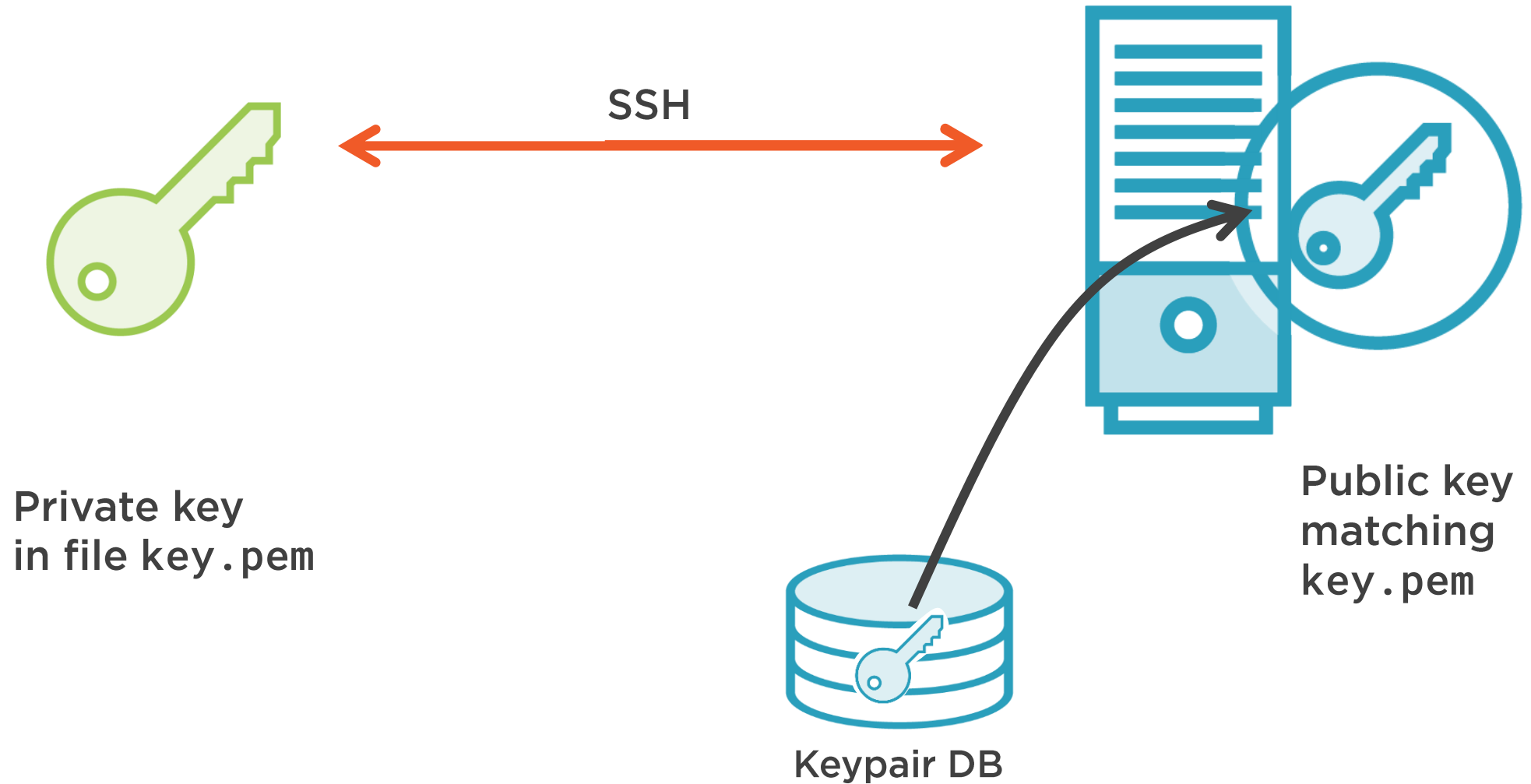
Console and network access

SSH keys

Personalizing the server:
Metadata and Userdata



SSH Keys



SSH Key Management: GUI

Launch Instance

[Details *](#)[Access & Security](#)[Networking *](#)[Post-Creation](#)[Advanced Options](#)

Key Pair ?

No key pairs available

Control access to your instance via key pairs, security groups, and other mechanisms.

Security Groups ?

☐ ssh

☐ default

openstack

demo

demo

Access & Security

[Security Groups](#)[Key Pairs](#)[Floating IPs](#)[API Access](#)

Filter

+ Create Key Pair

Import Key Pair

Key Pair Name	Fingerprint	Actions
No items to display.		
Displaying 0 items		



SSH Key Management: Command Line

```
$ openstack keypair create NAME > key.pem
```

```
$ chmod 600 key.pem
```

Existing key file

```
$ openstack keypair create --public-key key.pub NAME
```

New private key file
created by Nova

```
$ openstack keypair show NAME
```

```
$ openstack server create .... --key-name NAME ...
```

```
$ nova keypair-add NAME > key.pem
```

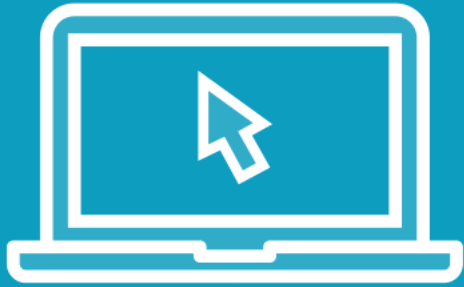
```
$ nova keypair-add --public-key key.pem NAME
```

```
$ nova keypair-show NAME
```

```
$ nova boot .... --key-name NAME ...
```



Demo



Key pairs



Personalization



What's in a server:
Overview of server concepts

Launching a server

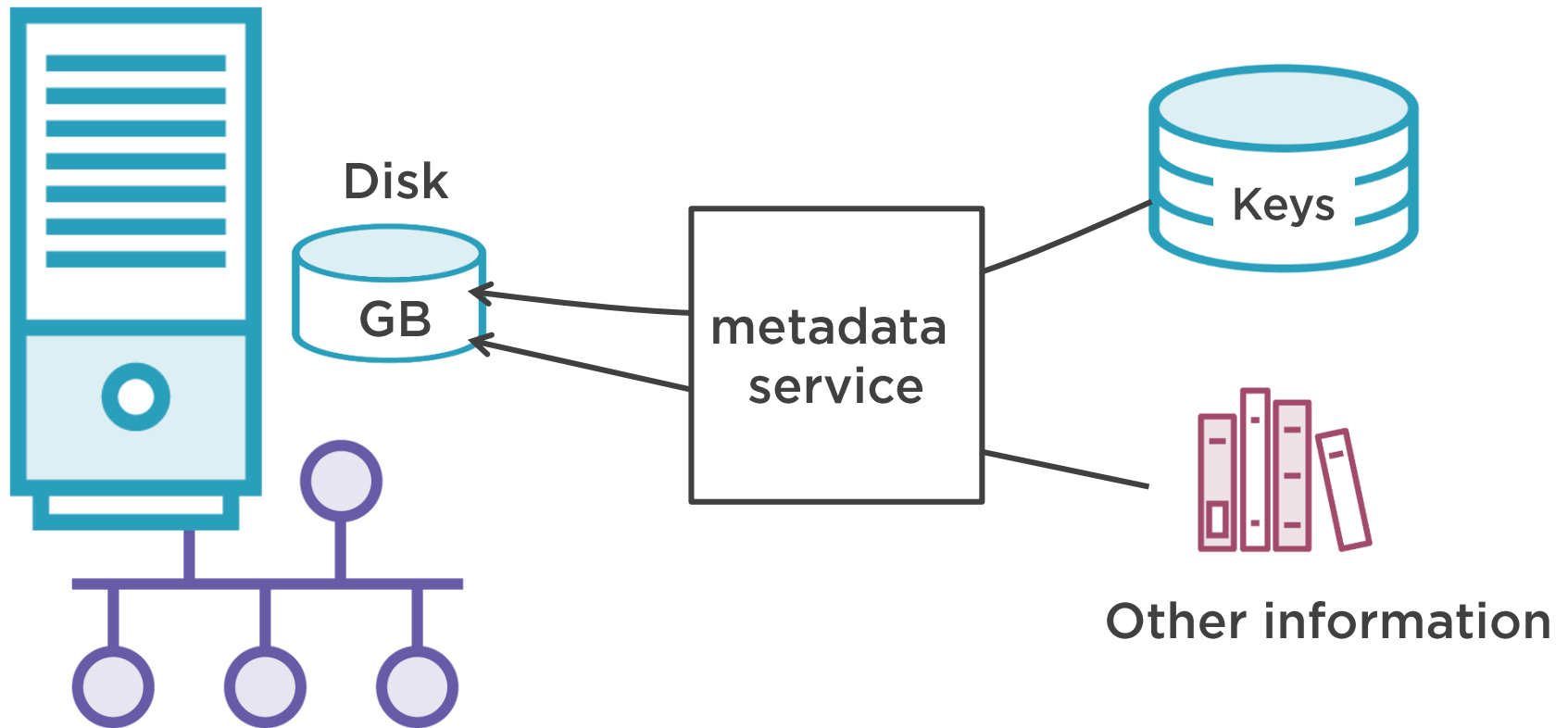
Console and network access

SSH keys

Personalizing the server:
Metadata and Userdata



Personalizing Your Instance



Examples of Instance Metadata



```
$ openstack server create \
  --property nservers=15
```

```
$ openstack server create \
  --user-data my-script.sh
```

```
$ openstack server create \
  --user-data my-config.yaml
```

```
$ openstack server create \
  --file /root/db.cfg=db.cfg
```

```
$ nova boot --meta KEY=VALUE
```

```
$ nova boot --user-data ...
```

```
$ nova boot --file ...
```

◀ Key-Value pairs

◀ Script (any scripting language)

◀ Cloud-config file

◀ File injection

◀ Nova client



User Data



cloud-init:

- retrieves user-data file from metadata service
- executes script
- or
- processes cloud-config file
- several other formats (gzip, MIME, ...)

metadata
service

```
#!/bin/bash
```

```
.....
```

```
#!/usr/bin/perl
```

```
...
```

```
#!/usr/bin/awk
```

```
.....
```

```
#cloud-config
```

```
.....
```

```
other formats
```

```
.....
```

Documented at <http://cloudinit.readthedocs.io>



cloud-config



Declarative

Describes the changes, not the steps for making changes



Written in YAML

YAML Ain't a Markup Language, common data declaration format



Simple

Faster to write, less errors than procedural scripts



cloud-config Examples

key

value

```
#cloud-config
timezone: "Asia/Tokyo"
```

```
#cloud-config
package_upgrade: true
```

Upgrades all packages
Works for all package managers

```
#cloud-config
```

```
packages:
```

- pwgen
- pastebinit
- - libpython2.7
- 2.7.3-0ubuntu3.1

Installs the packages
Specific version of libpython
Works for all package managers

```
#cloud-config
```

```
ntp:
```

```
servers:
```

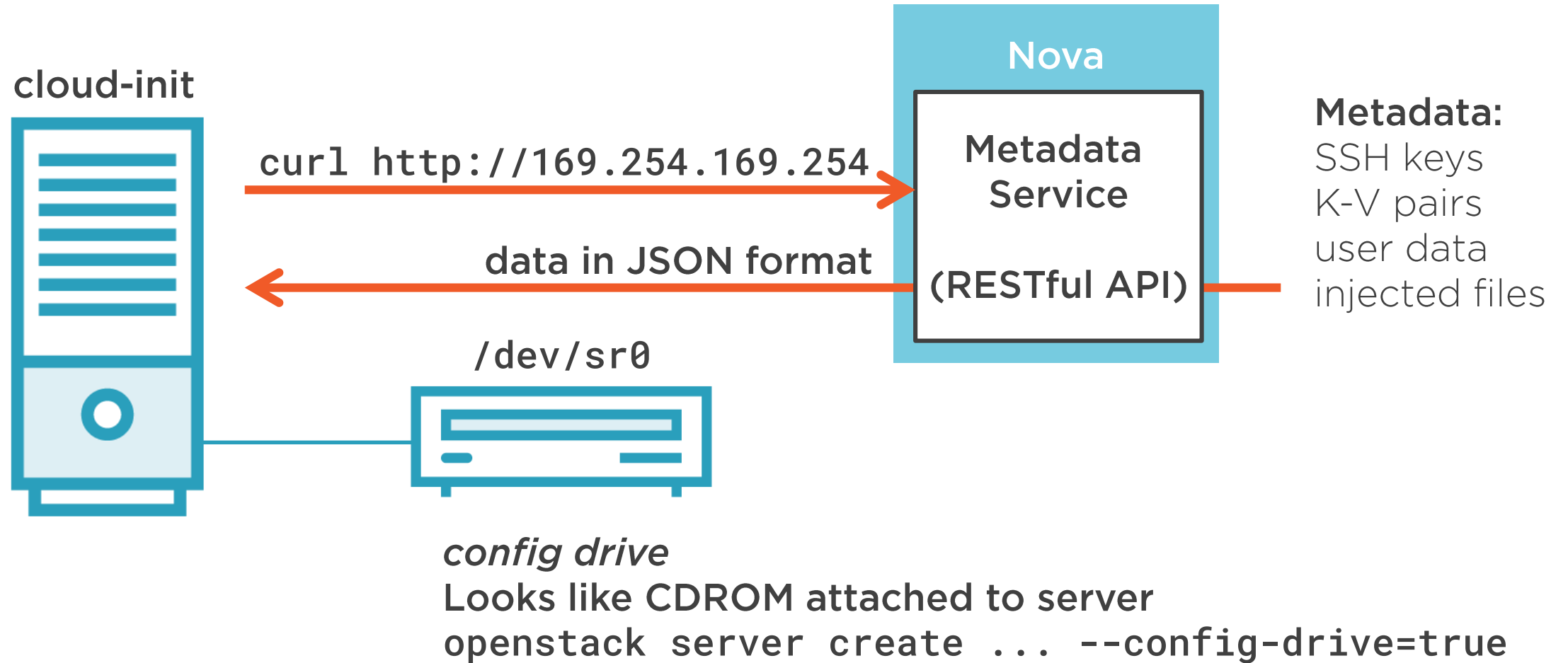
- my.ntp.server.local
- ntp.ubuntu.com
- 192.168.23.2

Installs NTP if necessary
Configures NTP with the above servers

map



How the Instance Retrieves Metadata



Demo



Key-value metadata

Access to the metadata service

User-data and cloud-config

Config-Drive



Summary



What's in a server

Launching one

- Images, flavors, snapshots

Accessing it

- SSH keys, security groups, floating IPs

Personalizing it

- cloud-config, metadata API

