CLOUD NETWORKING WORKSHOP

David Casper
Moogsoft

Chris Swan
CohesiveFT

AGENDA

- ‘EC2 Classic’ networks – NAT and Firewalls
  - Start up a simple instance
  - Connect over SSH
  - Expose a web server
- Virtual Private Clouds
  - Public and private networks
  - Subnets
  - Accessing resources on a private subnet
- Overlay networks
  - Container internetworking

  - Building networks across clouds
  - Securing data in motion
  - Edge connectivity to data centers
  - Network application services in containers

GO AT YOUR OWN PACE

Detailed instructions (and these slides) are available at:

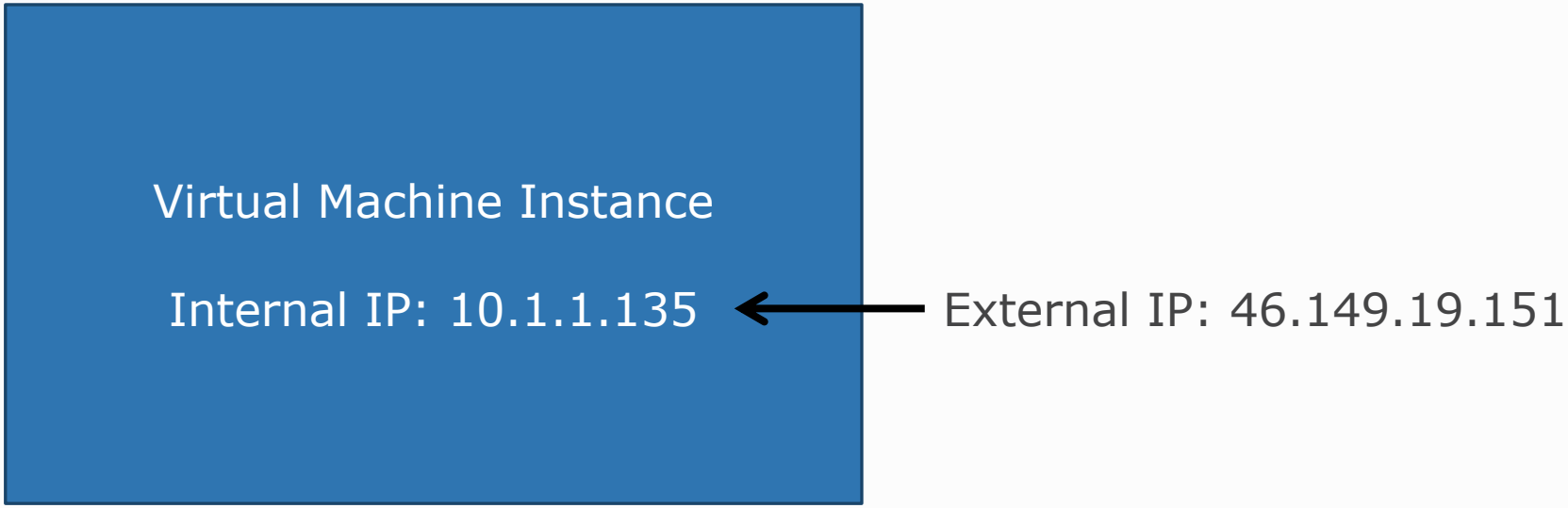
https://github.com/cpswan/cloud-networking-workshop

is.gd/odcacw

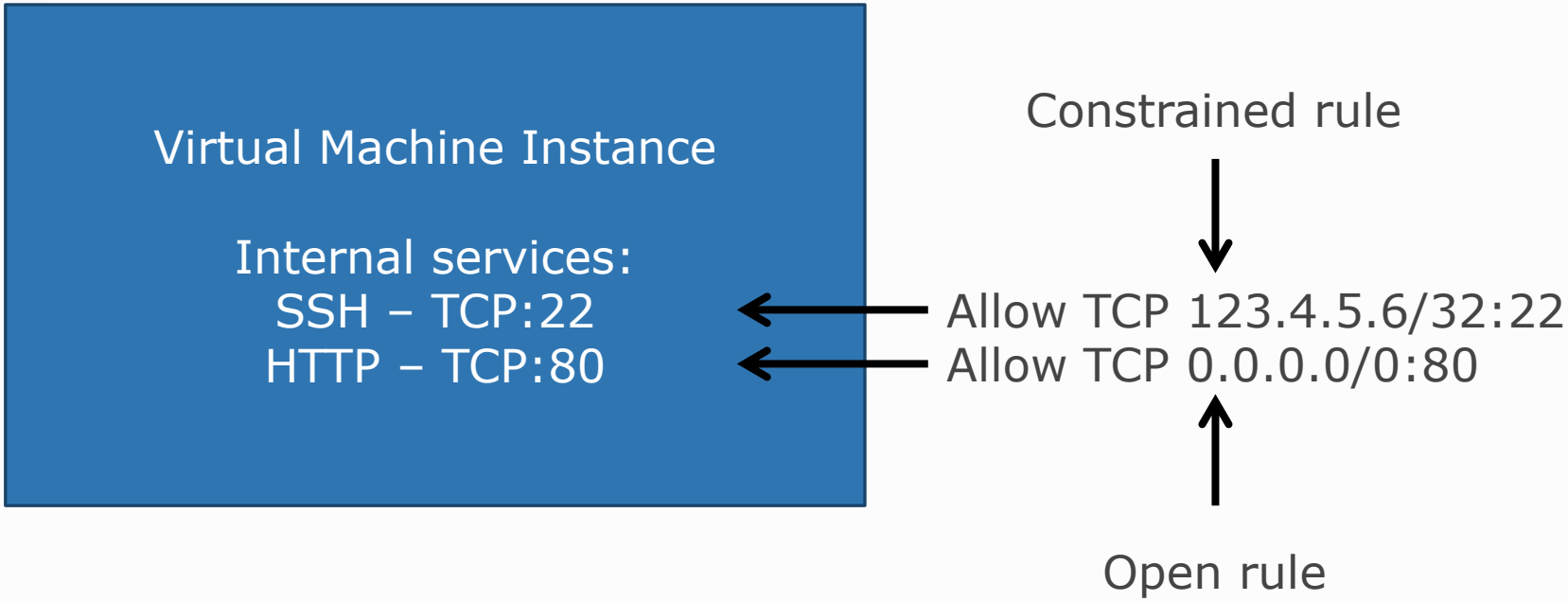
‘EC2 CLASSIC’ NETWORKS

NAT and Firewalls

NAT


Virtual Machine Instance
Internal IP: 10.1.1.135
External IP: 46.149.19.151

FIREWALLS (SECURITY GROUPS)



LAUNCH AN INSTANCE

1) console.aws.amazon.com

2) Sign In or Create an AWS Account

You may sign in using your existing Amazon.com account or you can create a new account by selecting "I am a new user."

3) Amazon Web Services

Compute & Networking

Direct Connect
Dedicated Network Connection to AWS

EC2
Virtual Servers in the Cloud

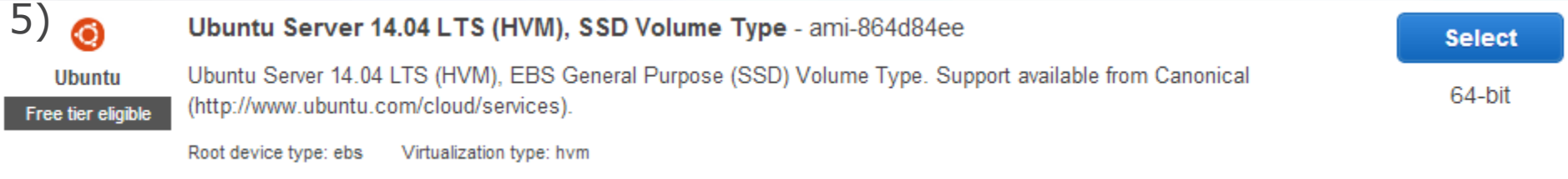
4) Create Instance

To start using Amazon EC2

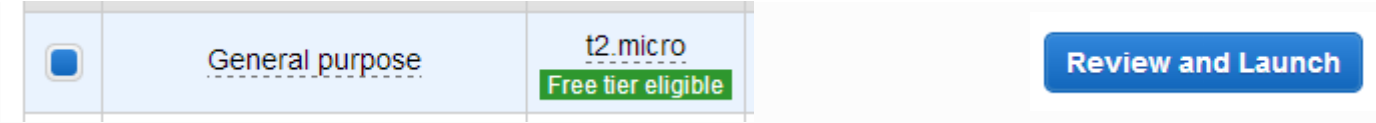
Launch Instance

LAUNCH AN INSTANCE CONT.

5)



6)



7) STOP! – Let's take a look at what's happening with the network

8) Go ahead and launch it, then go to the instances view to see private and public IP addresses

CONNECT ON SSH

Connect

```
ssh -i my_key.pem ubuntu@public.ip
```

Show instance view of IP

```
ubuntu@public.ip:~$ ifconfig
```

Show NAT IP

```
ubuntu@public.ip:~$ curl ifconfig.co
```

INSTALL A WEB SERVER

```
ubuntu@public.ip:~$ sudo apt-get install -y nginx
```

Confirm that it works:

```
ubuntu@public.ip:~$ curl localhost
```

Now browse to the public IP of the instance

Something's not right ☹

ADD WEB SERVER FIREWALL RULE

1) NETWORK & SECURITY
Security Groups

2) launch-wizard-1

3) Description | Inbound
Edit

4) Add Rule

5)

Edit inbound rules

Type	Protocol	Port Range	Source	
SSH	TCP	22	Anywhere 0.0.0.0/0	
Custom TCP Rule	TCP	0	Custom IP	

Custom TCP Rule
Custom UDP Rule
Custom ICMP Rule
Custom Protocol
All TCP
All UDP
All ICMP
All traffic
SSH
SMTP
DNS
HTTP
POP3
IMAP

Cancel Save

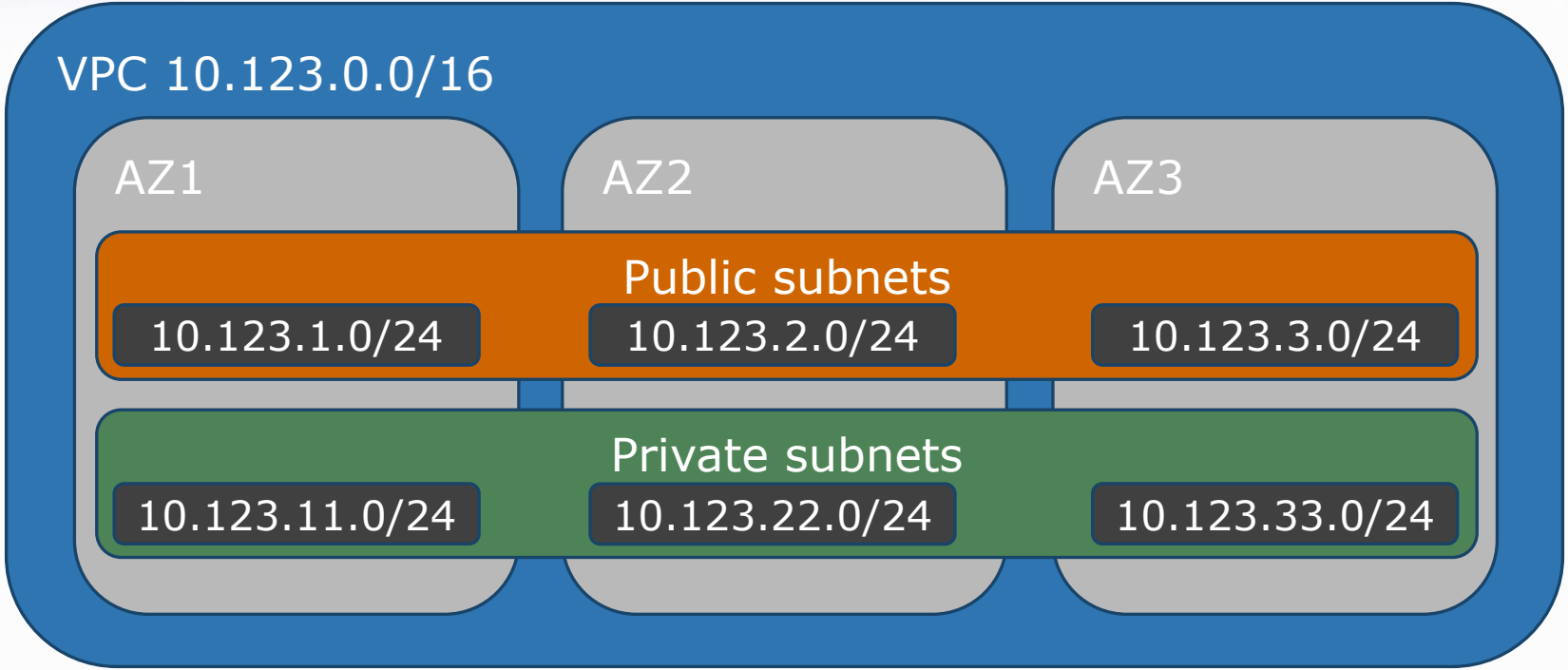
1db89d78

SUCCESS


Welcome to nginx!
public.ip
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.

VIRTUAL PRIVATE CLOUDS

VPC OVERVIEW



HAVE A GO AT CREATING A VPC

1) Start VPC Wizard

2)

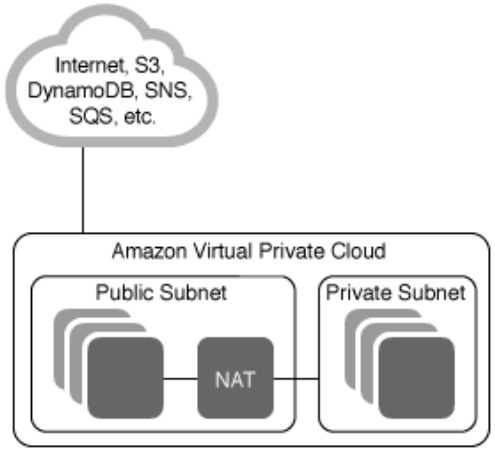
- VPC with a Single Public Subnet
- **VPC with Public and Private Subnets**
- VPC with Public and Private Subnets and Hardware VPN Access
- VPC with a Private Subnet Only and Hardware VPN Access

In addition to containing a public subnet, this configuration adds a private subnet whose instances are not addressable from the Internet. Instances in the private subnet can establish outbound connections to the Internet via the public subnet using Network Address Translation (NAT).

Creates:

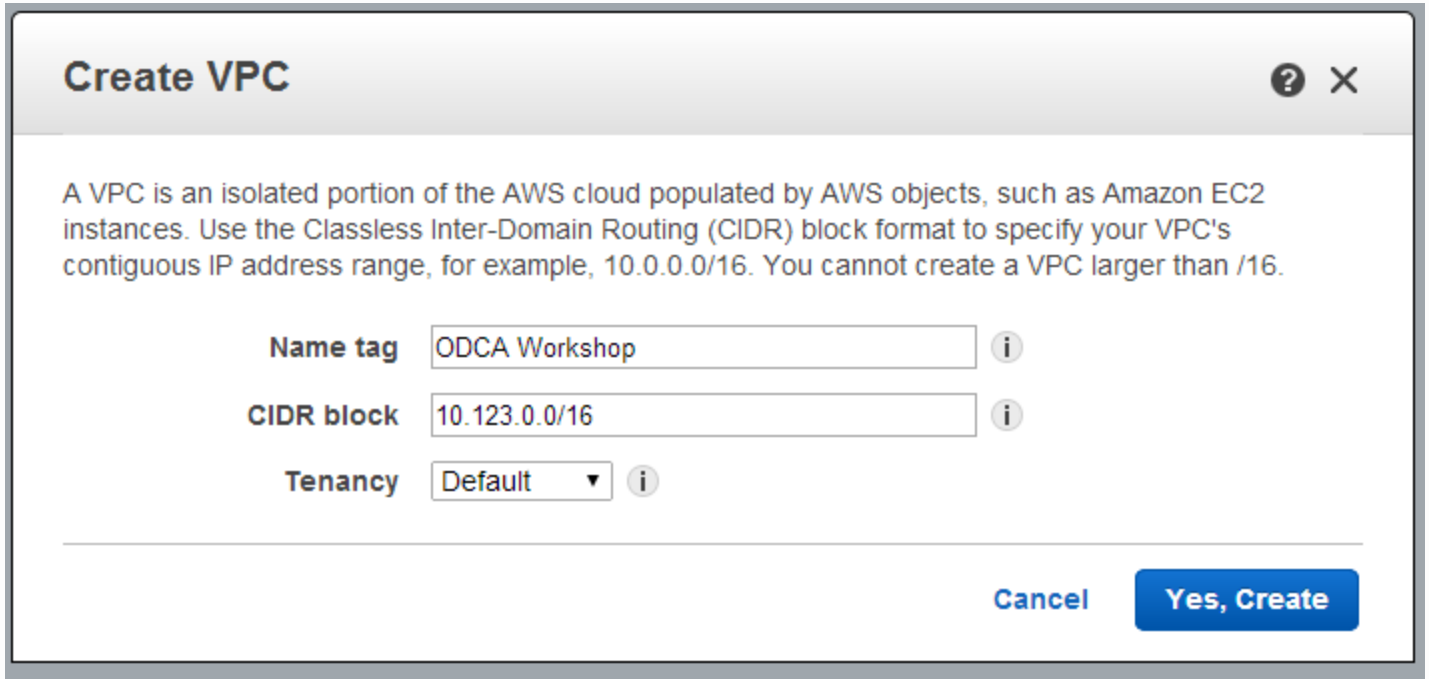
A /16 network with two /24 subnets. Public subnet instances use Elastic IPs to access the Internet. Private subnet instances access the Internet via a Network Address Translation (NAT) instance in the public subnet. (Hourly charges for NAT instances apply.)

Select



WIZARD WON'T DO ALL SUBNETS

So it might be just as easy to create VPC by hand and add subnets:



CREATING SUBNETS

NB: nothing here about public or private!

Create Subnet

Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.

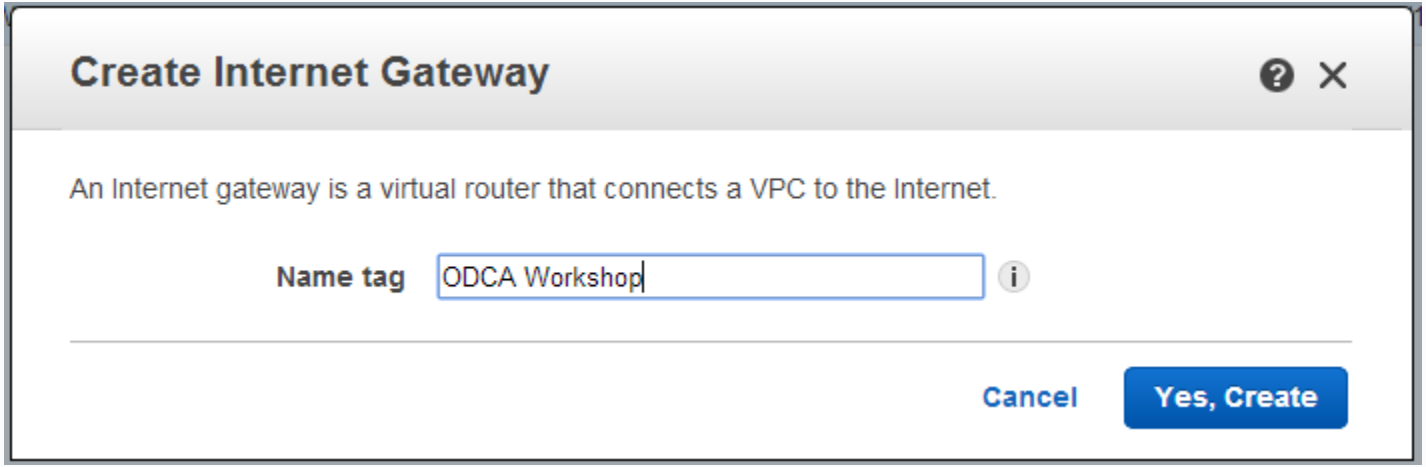
Name tag	ODCA-Pub-1A
VPC	vpc-25229340 (10.123.0.0/16) | ODCA Workshop
Availability Zone	us-east-1a
CIDR block	10.123.1.0/24

Cancel | Yes, Create

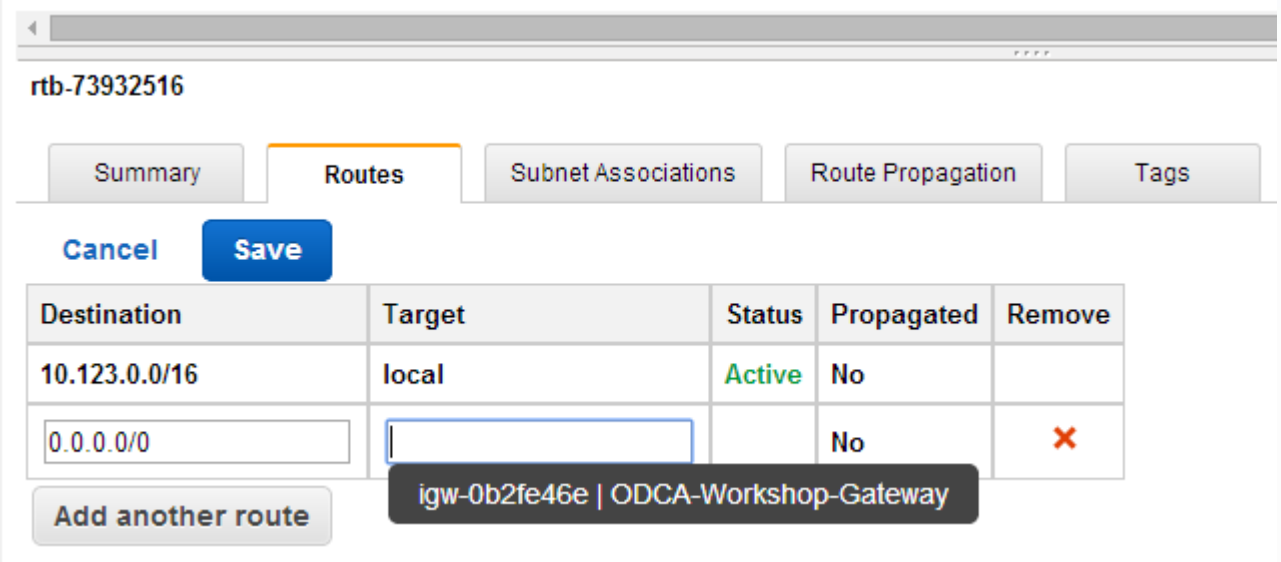
INTERNET GATEWAY

1) Create Internet Gateway

2)



ROUTING TO THE INTERNET



MAKING SUBNETS PRIVATE

Create Route Table | Delete Route Table | Set As Main Table

Search Route Tables and their properti

Name	Route Table ID	Associated With
ODCA Default	rtb-73932516	0 Subnets
ODCA Private	rtb-41239524	3 Subnets

rtb-41239524 | ODCA Private

Summary | Routes | Subnet Associations | Rou

Edit

Subnet	CIDR
subnet-5905f500 (10.123.11.0/24) | ODCA-Priv-1A	10.123.11.0/24
subnet-3dabbb15 (10.123.22.0/24) | ODCA-Priv-1C	10.123.22.0/24
subnet-f164b286 (10.123.33.0/24) | ODCA-Priv-1D	10.123.33.0/24

The following subnets have not been associated with any route tables and are therefore using the main table routes:

Subnet	CIDR
subnet-70ef1f29 (10.123.1.0/24) | ODCA-Pub-1A	10.123.1.0/24
subnet-2cabbb04 (10.123.2.0/24) | ODCA-Pub-1C	10.123.2.0/24
subnet-0364b274 (10.123.3.0/24) | ODCA-Pub-1D	10.123.3.0/24

AUTO IP ASSIGNMENT

Modify Auto-Assign Public IP

Enable auto-assign public IP to automatically request a public IP address for instances launched into this subnet.

☑ Enable auto-assign Public IP

Note: You can override the auto-assign public IP setting for each individual instance at launch time. Regardless of how you've configured the auto-assign public IP feature, you can assign a public IP address to an instance that has a single, new network interface with a device index of eth0.

Cancel Save

NOW LAUNCH AN INSTANCE INTO ODCA WORKSHOP VPC PUBLIC NET

1)

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, req
management role to the instance, and more.

Number of instances	1
Purchasing option	☐ Request Spot Instances
Network	vpc-25229340 (10.123.0.0/16) | ODCA Workshop
Subnet	subnet-70ef1f29(10.123.1.0/24) | ODCA-Pub-1A | u 251 IP Addresses available
Auto-assign Public IP	Use subnet setting (Enable)

2) **Review and Launch**

REPEAT PREVIOUS STEPS

Connect

```
ssh -i my_key.pem ubuntu@public.ip
```

Show instance view of IP

```
ubuntu@public.ip:~$ ifconfig
```

Show NAT IP

```
ubuntu@public.ip:~$ curl ifconfig.co
```

Install web server

```
ubuntu@public.ip:~$ sudo apt-get install -y nginx
```

Confirm that it works:

```
ubuntu@public.ip:~$ curl localhost
```

Add HTTP to security group and browse to the public IP of the instance

NOW LAUNCH AN INSTANCE INTO ODCA WORKSHOP VPC PRIVATE NET

1) Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, rec
management role to the instance, and more.

Number of instances	1
Purchasing option	☐ Request Spot Instances
Network	vpc-25229340 (10.123.0.0/16) | ODCA Workshop
Subnet	subnet-3dabbb15(10.123.22.0/24) | ODCA-Priv-1C 250 IP Addresses available
Auto-assign Public IP	Use subnet setting (Disable)

2) **Review and Launch**

USE THE PUBLIC FACING INSTANCE AS AN SSH JUMP BOX

Set up a connection that provides a tunnel to the private IP

```
ssh -i my_key.pem -L 2222:private.ip:22 \
ubuntu@public.ip
```

Then use that tunnel to connect (in a different tab/window)

```
ssh -i my_key.pem -p 2222 ubuntu@localhost
```

CONNECTING TO A PRIVATE SUBNET SERVICE

Start up a simple Python webserver on the private instance:

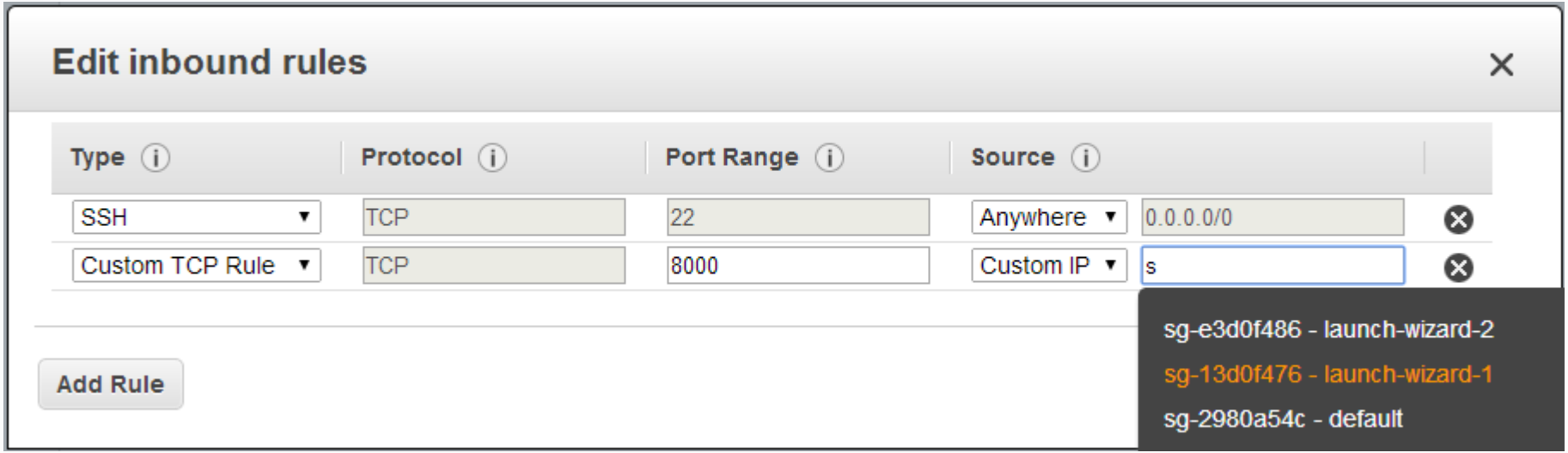
```
ubuntu@private.ip:~$ echo 'Hello ODCA' > index.html
ubuntu@private.ip:~$ python -m SimpleHTTPServer
```

Now connect to it from the public instance:

```
ubuntu@public.ip:~$ curl private.ip:8000
```

It won't work – as port 8000 needs to be allowed in the security group

OPEN SECURITY GROUP TO ALLOW PRIVATE SERVICE TO BE ACCESSED



SUCCESS ☺

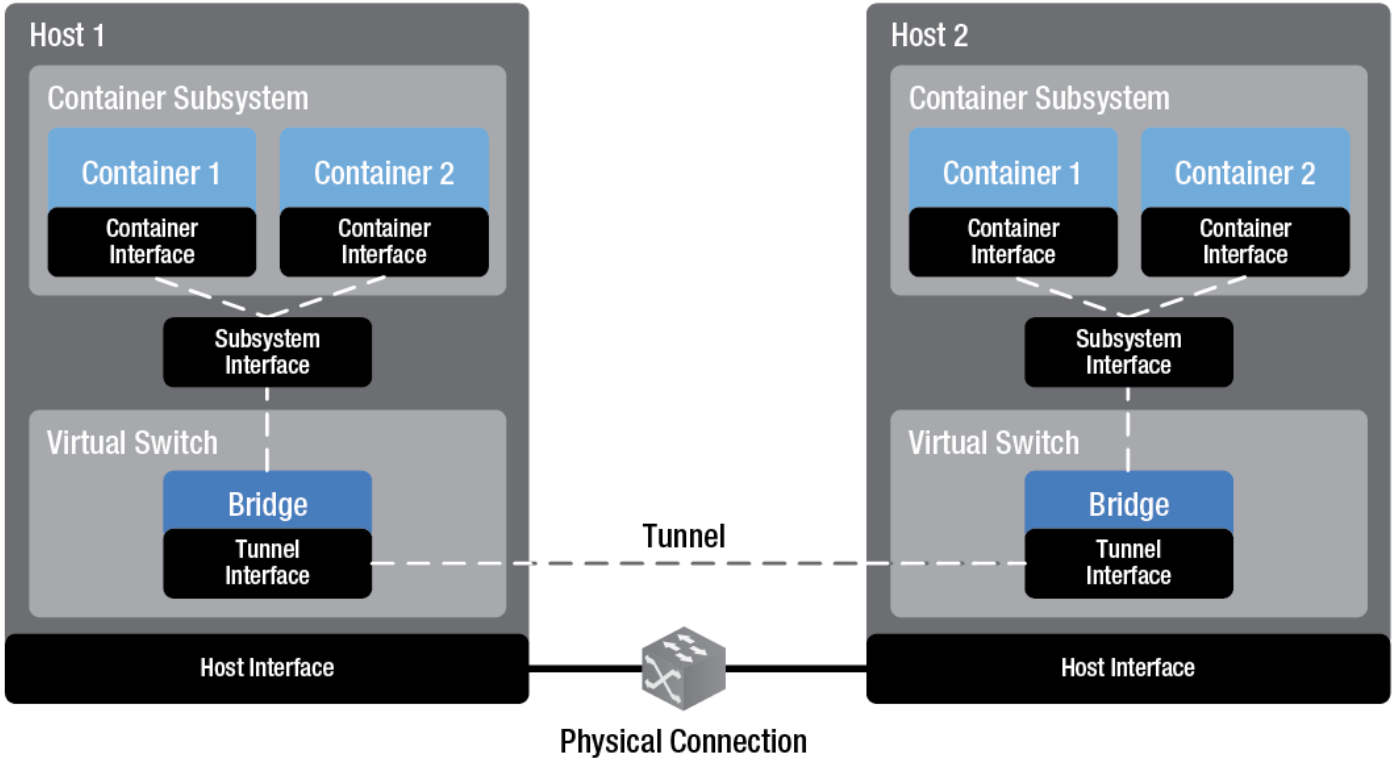
```
ubuntu@public:~$ curl private.ip:8000
Hello ODCA
ubuntu@public:~$
```

WE SKIPPED OVER A FEW THINGS

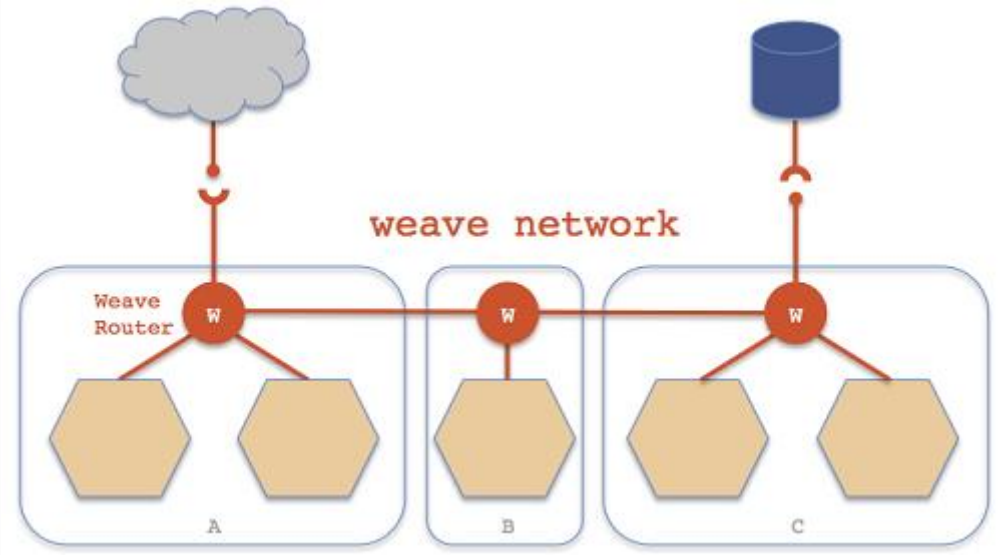
- NAT instances – so that private subnets can reach the outside world
- Elastic IPs
- Network ACLs (which work in addition to security groups)
- Multi-homing (network interfaces)
- Load Balancers
- VPC peering
- VPN gateways
- Direct connect

OVERLAY NETWORKS

SDN UM V.2 USAGE SCENARIO 4



USING WEAVE



CREATE WEAVE SECURITY GROUP

Create Security Group

Security group name	Weave
Description	UDP and TCP 6783
VPC	vpc-1df54678 (10.123.0.0/16) | ODCA Workshop

CONFIGURE WEAVE SECURITY GROUP

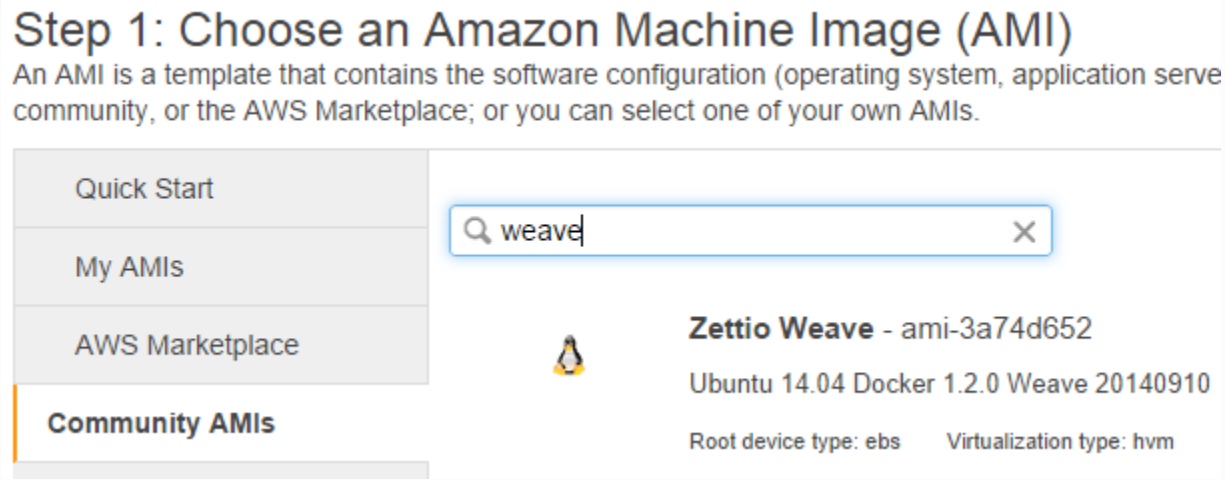
Edit inbound rules

Type	Protocol	Port Range	Source	
SSH	TCP	22	Anywhere	0.0.0.0/0
Custom TCP Rule	TCP	6783	Custom IP	sg-ca1d3baf
Custom UDP Rule	UDP	6783	Custom IP	sg

sg-ca1d3baf - Weave
sg-911f39f4 - default

Add Rule

START 2 INSTANCES OF WEAVE



CHOOSING THE WEAVE SECURITY GROUP ALONG THE WAY

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this
allow Internet traffic to reach your instance, add rules that allow unrestricted access to t
Amazon EC2 security groups.

Assign a security group: ○ Create a **new** security group

◉ Select an **existing** security group

	Security Group ID	Name
☐	sg-911f39f4	default
☑	sg-ca1d3baf	Weave

START A CONTAINER ON WEAVE 1

Connect

```
ssh -i my_key.pem ubuntu@weave1.public.ip
```

Become root

```
ubuntu@weave1:~$ sudo su
```

Start Weave

```
ubuntu@weave1:/home/ubuntu# weave launch 10.0.0.1/16
```

Start a container using Weave

```
ubuntu@weave1:/home/ubuntu# C=$(weave run
10.0.1.1/24 -it ubuntu)
```

START A SERVICE IN THE CONTAINER ON WEAVE 1

Connect to the container (hit return twice to get prompt)

```
ubuntu@weave1:/home/ubuntu# docker attach $C
```

Create and start a service

```
root@0123456789ab:/#  echo 'Hello ODCA' > index.html
root@0123456789ab:/#  python3 -m http.server
```

START A CONTAINER ON WEAVE 2

Connect

```
ssh -i my_key.pem ubuntu@weave2.public.ip
```

Become root

```
ubuntu@weave2:~$ sudo su
```

Start Weave

```
ubuntu@weave2:/home/ubuntu# weave launch 10.0.0.1/16
weave1.private.ip
```

Start a container using Weave

```
ubuntu@weave2:/home/ubuntu# C=$(weave run
10.0.1.2/24 -it ubuntu)
```

CREATE A CLIENT IN THE CONTAINER ON WEAVE 2

Connect to the container (hit return twice to get prompt)

```
ubuntu@weave2:/home/ubuntu# docker attach $C
```

Do a quick ping test

```
root@ba9876543210:/#  ping -c 3 10.0.1.1
```

Install curl

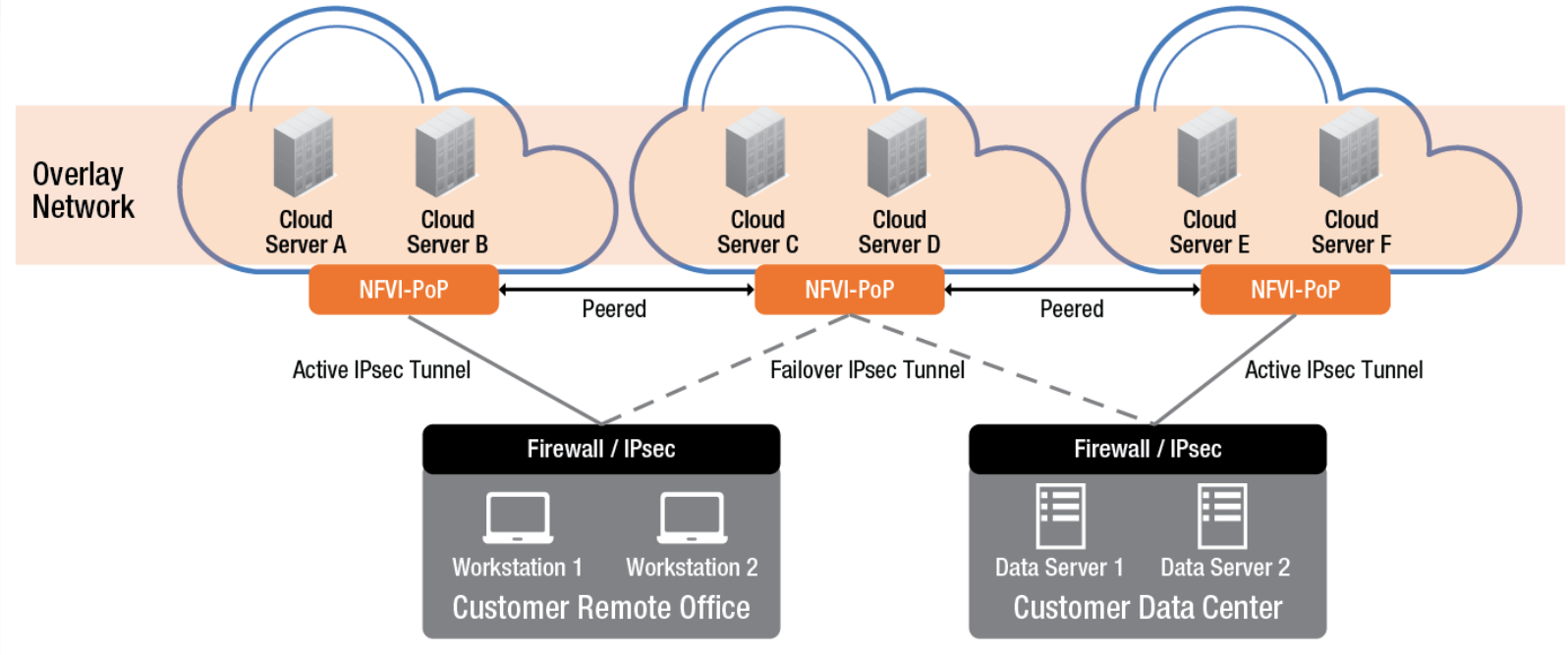
```
root@ba9876543210:/#  apt-get update
root@ba9876543210:/#  apt-get install -y curl
```

Access the service on Weave 1

```
root@ba9876543210:/#  curl 10.0.1.1:8000
```

OVERLAY NETWORKS CONT

SDN UM V.2 USAGE SCENARIO 8



NO BLOW BY BLOW SCREEN SHOTS

If you've got this far then you're probably ahead of the group. Follow the detailed instructions on the Github Wiki to:

- Launch and configure a pair of VNS3 network managers (one in US-East and the other in US-West)
- Connect a client VM into the secured overlay network
- Deploy a container with SSL termination and load balancing

FORECAST 2014

ACCELERATE YOUR ENTERPRISE CLOUD INNOVATION

OPEN DATA CENTER ALLIANCE

Q&A