Deloitte.

AWS Training Workshop Guide

18 October 2018

Session 1 - Introduction

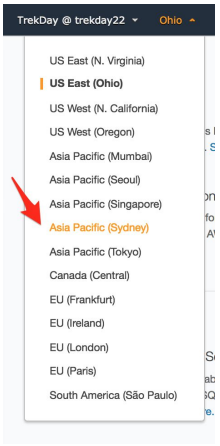
In this session we will get introduced to AWS and create our first virtual server.

Take the following steps, one at a time. There is plenty of time to complete the session and ask any questions along the way.

Getting Started

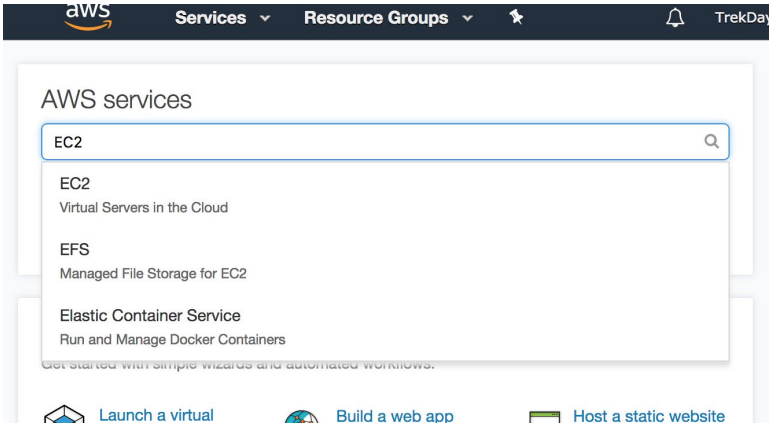
- Log into the AWS console using the details provided on your Credentials Note

- Set the region to Sydney (stay in this region today, check from time to time)

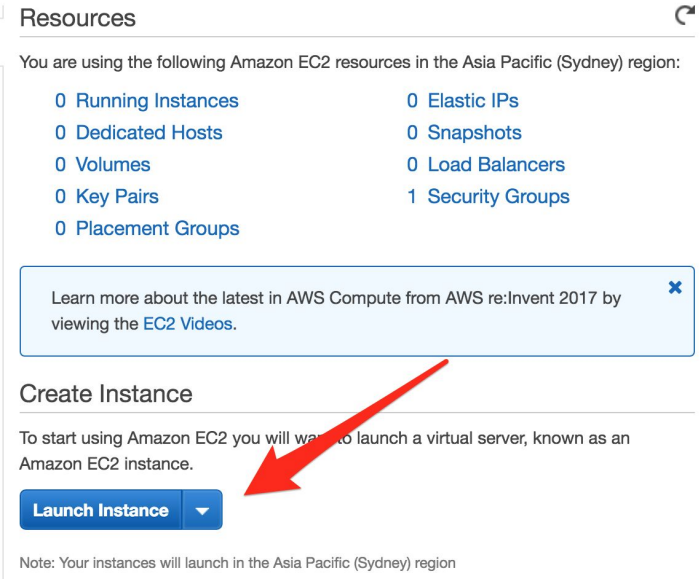


Create a Virtual Server - EC2 Instance

- Navigate to EC2



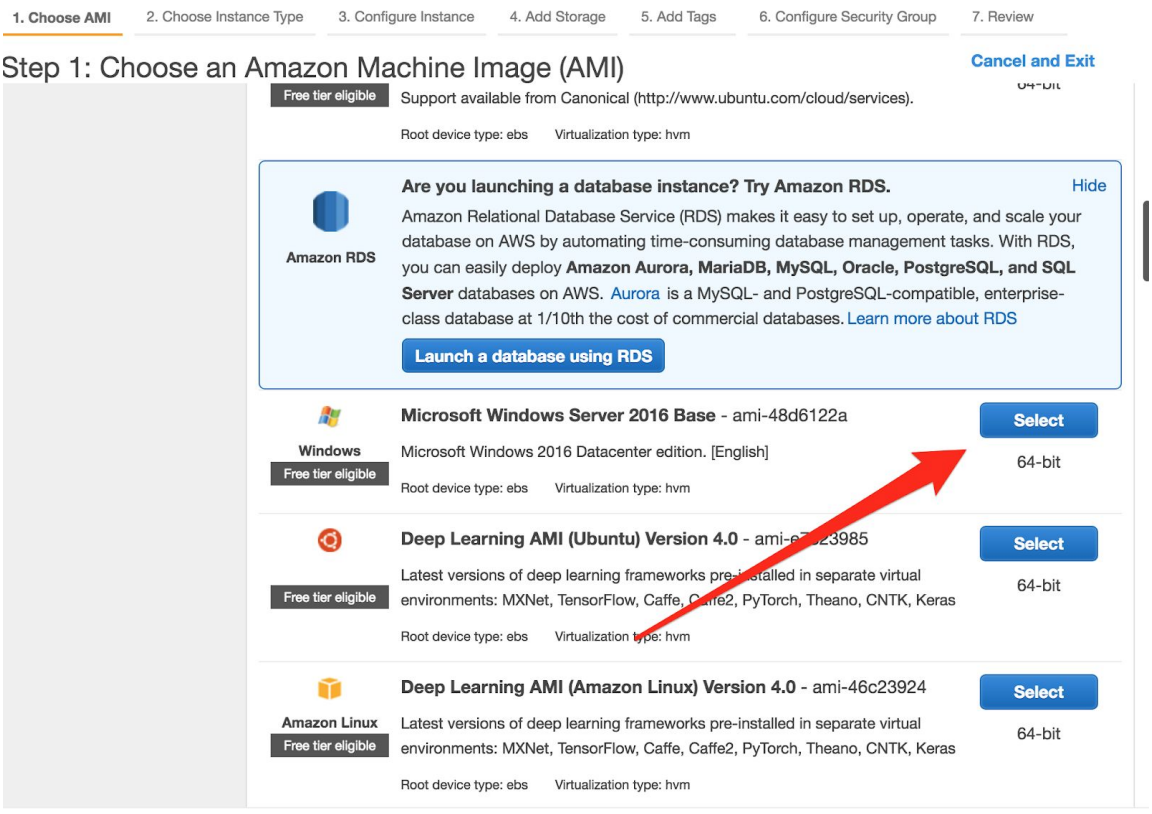
- Click “Launch Instance”



- Step 1: Select an Amazon Machine Image (AMI)

Take a look at all the different Amazon Machine Images that are available. There are many variants of Windows and Linux.

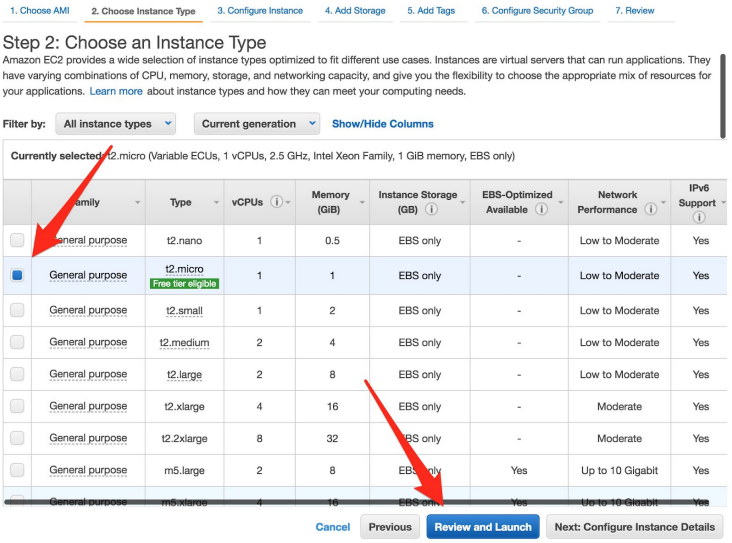
- Select "Microsoft Windows Server 2016 Base"



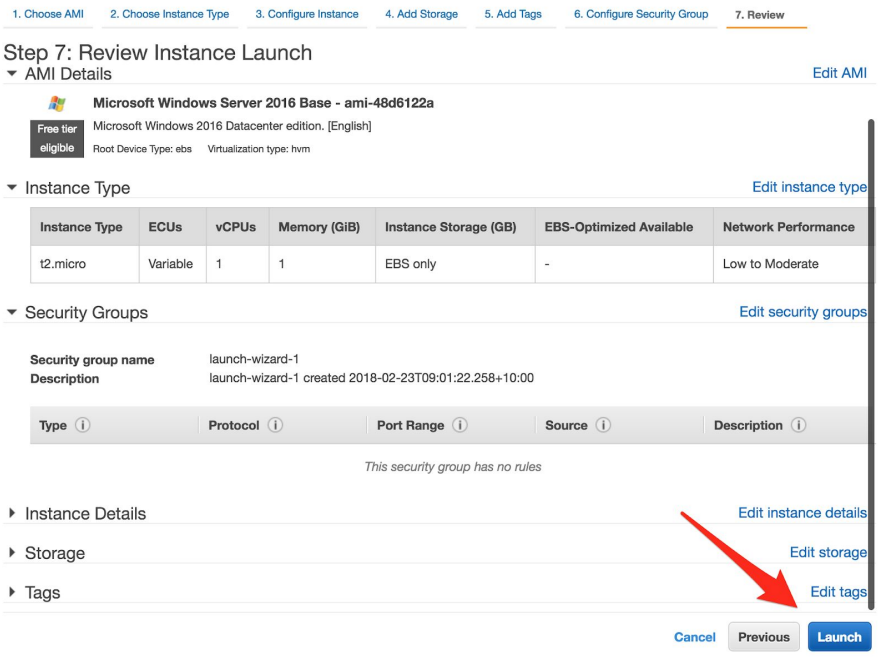
- Step 2: Choose an Instance Type

Take a look at all the different Instance Types available. There are sizes for almost any workload.

- Select "t2.micro"

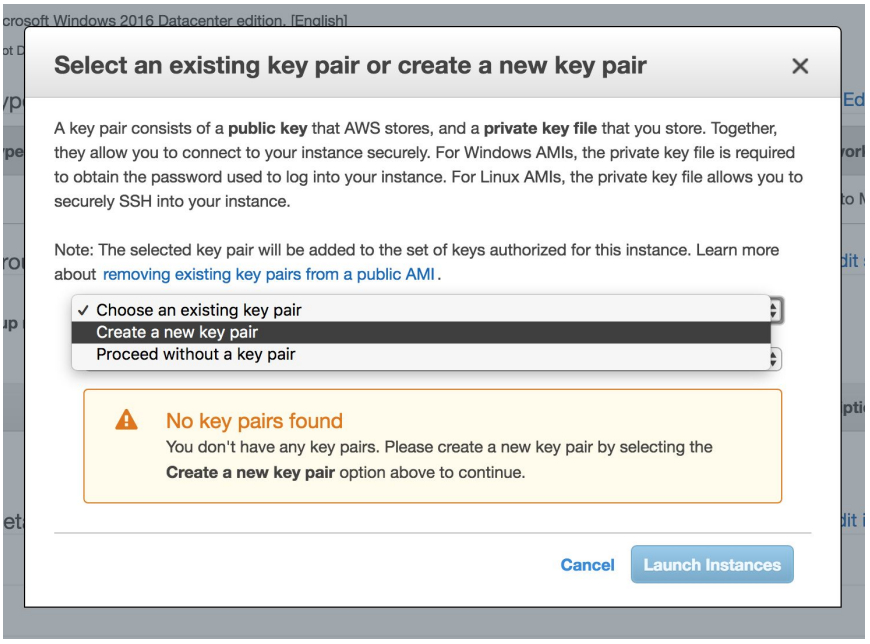


- Select “Review and Launch”, then “Launch”



- Popup: "Select an existing key pair or create a new key pair"

- Select: "Create a new key pair"



- Provide: Type a name, then "Download Key Pair" and keep it safe, we will use this a couple of times today.

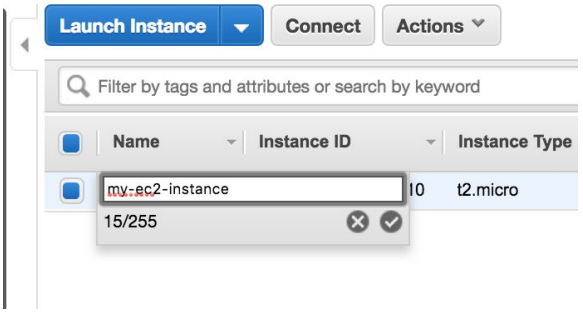
- Select: "Launch instances"

- Scroll down and select "View Instances"

- Review the "Instances" page.

This is where you will see all the instances you have launched. We will be coming back to this list several times today.

- Set the name of the instance, by clicking the pencil in the Name column. Set the name to "my-ec2-instance".



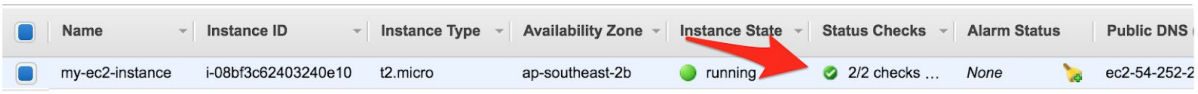
Tip: Make sure you select the 'tick' when changing the name, else it won't take.

- Review the "Description" tab for the running instance.

- Look at the "Tags" tab, notice that the name is set as a tag.

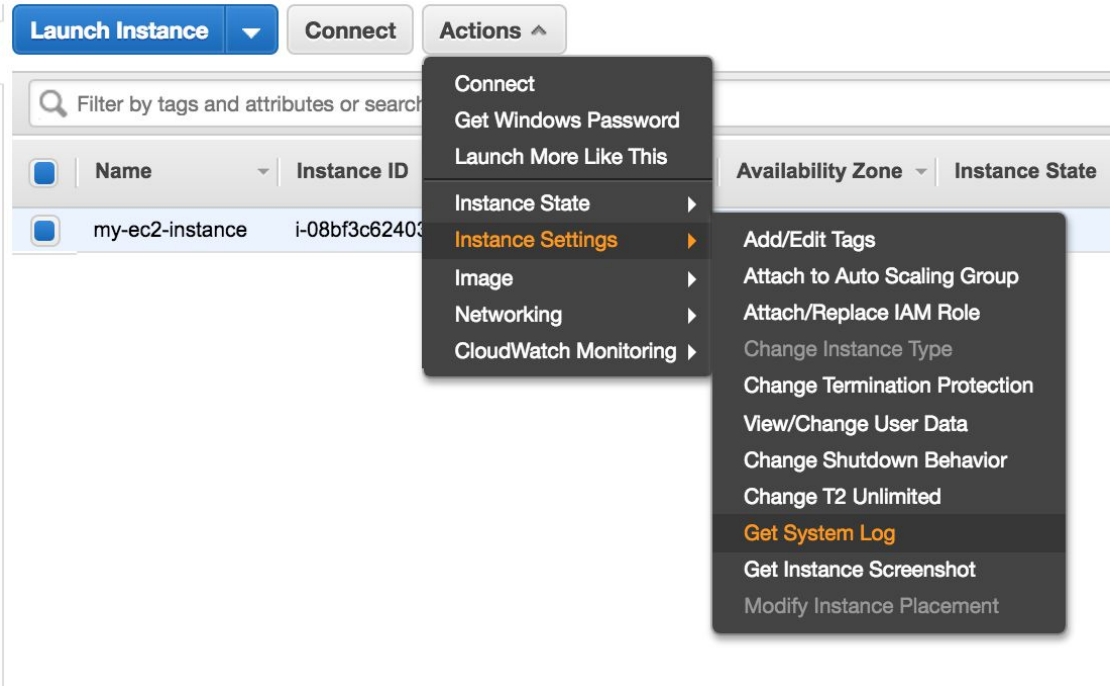
Connect to the Virtual Server - EC2 Instance

- Wait for the "Status Checks" 2/2. This means that the host and guest OS's are passing status checks.

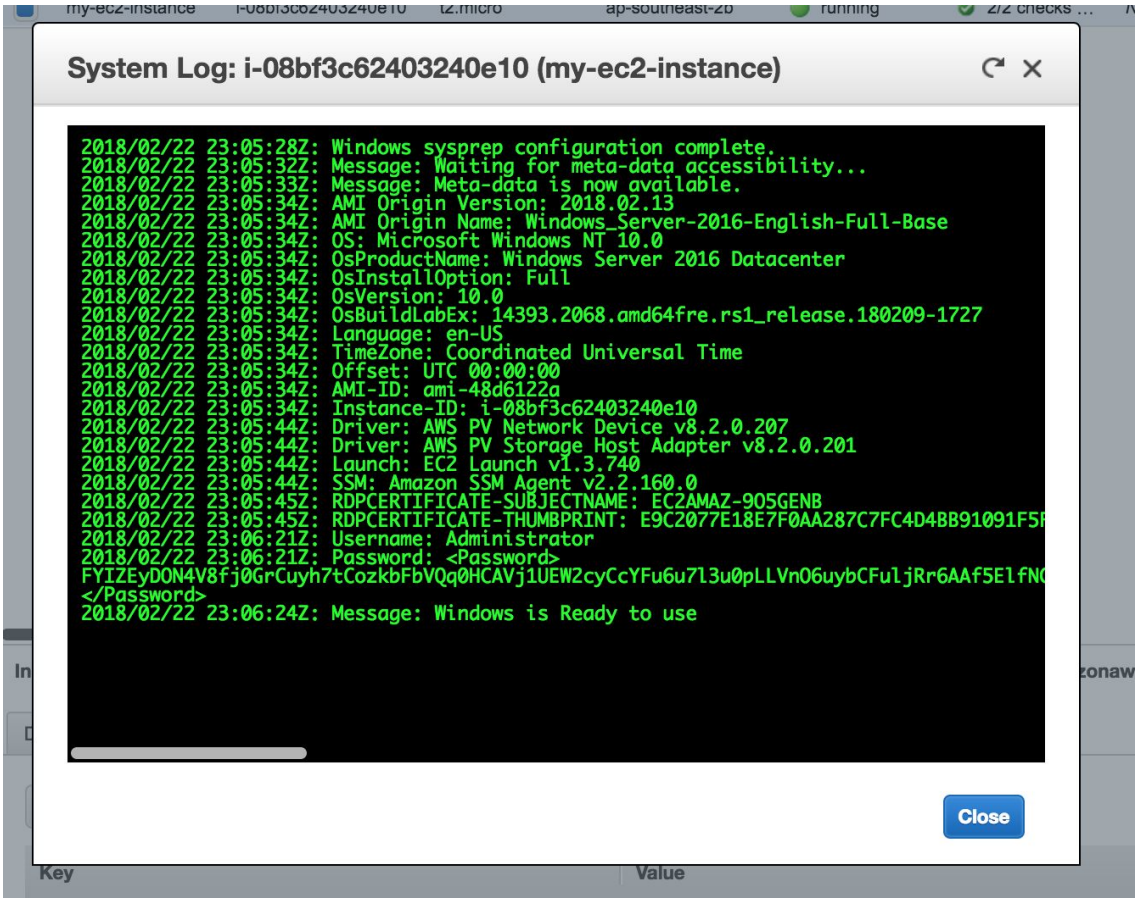


- Let's check that the Windows server is up and running:

- With the instance selected, select "Actions", "Instance Settings", "Get System Log"



- You should be able to see system logs sent from the server.

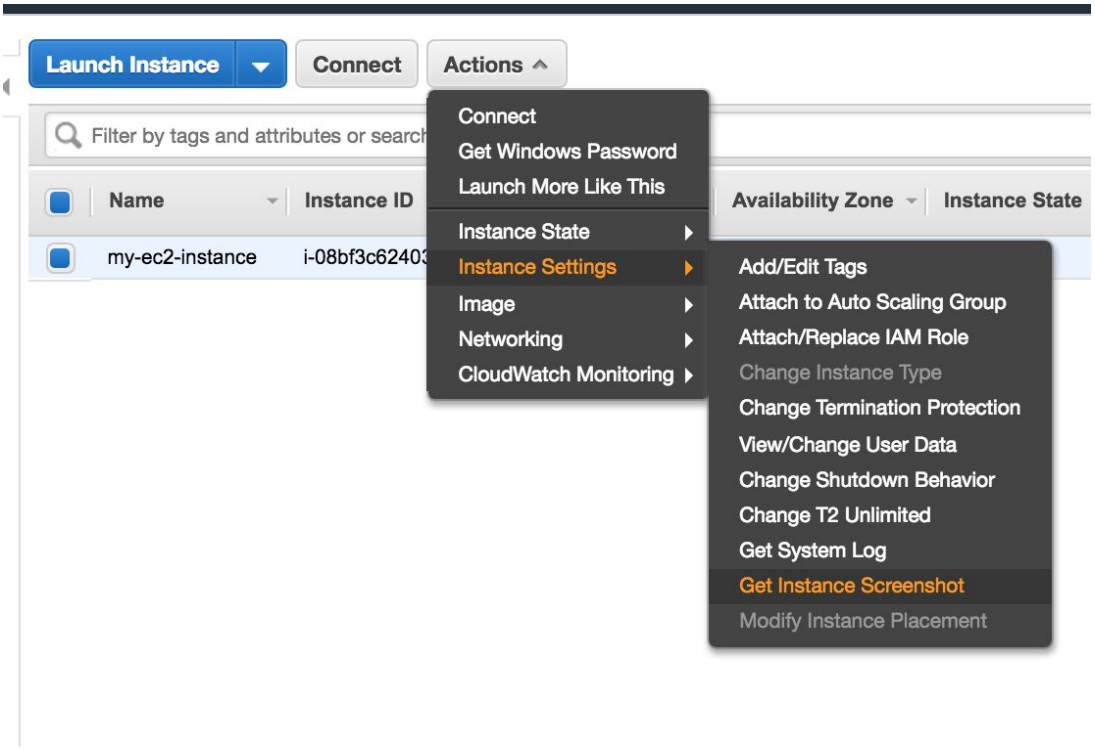


- Notice the Password field is encrypted

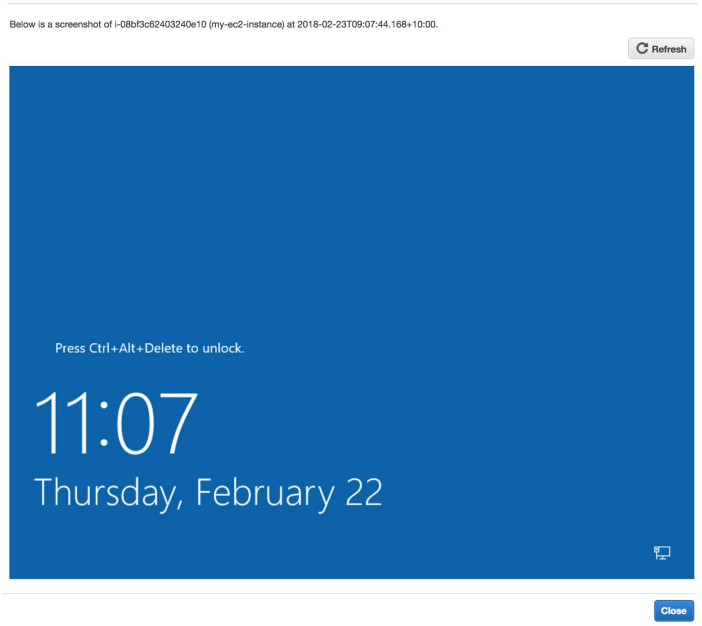
- Is there a line that says: "Message: Windows is Ready to use"?

- Select "Close"

- With the instance selected, select "Actions", "Instance Settings", "Get Instance Screenshot":



- You should see an image of the screen output of the server.



- You can't interact with this image, so lets log in to the server.

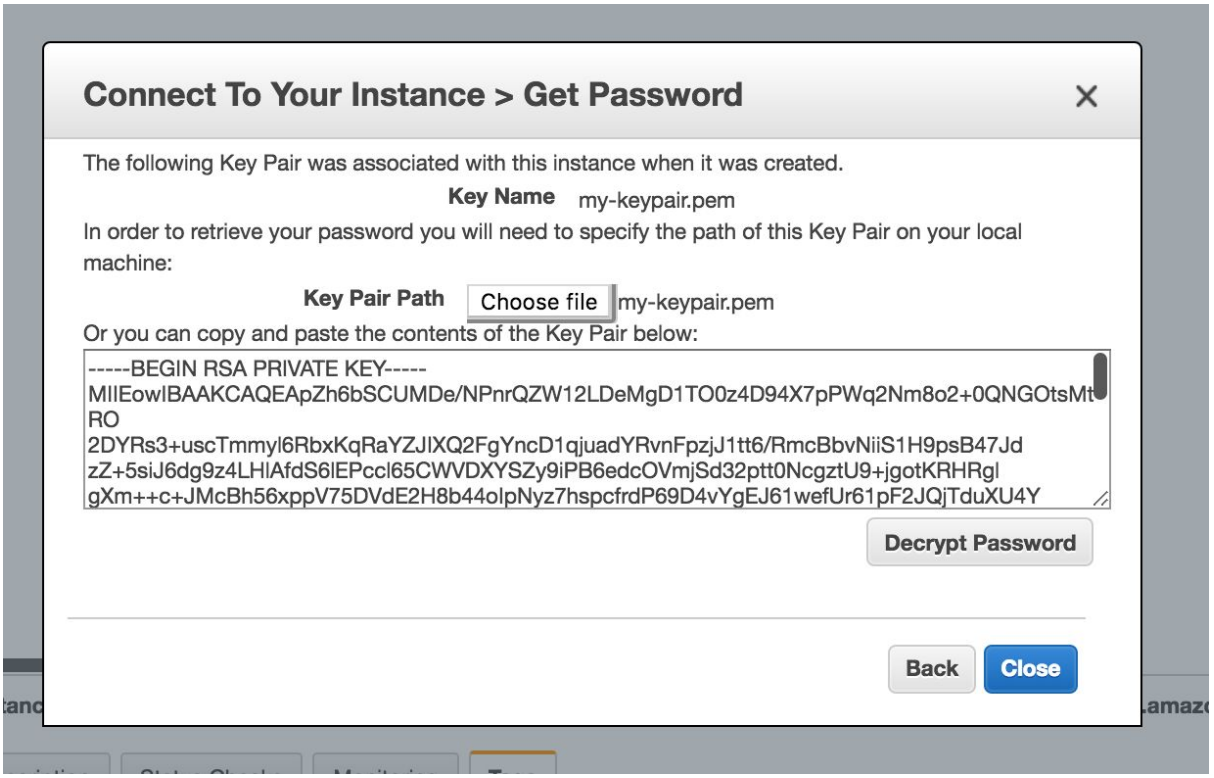
- Select “Close”

- Let’s decrypt the Windows Administrator password:

- With the instance selected, select "Actions", "Get Windows Password"

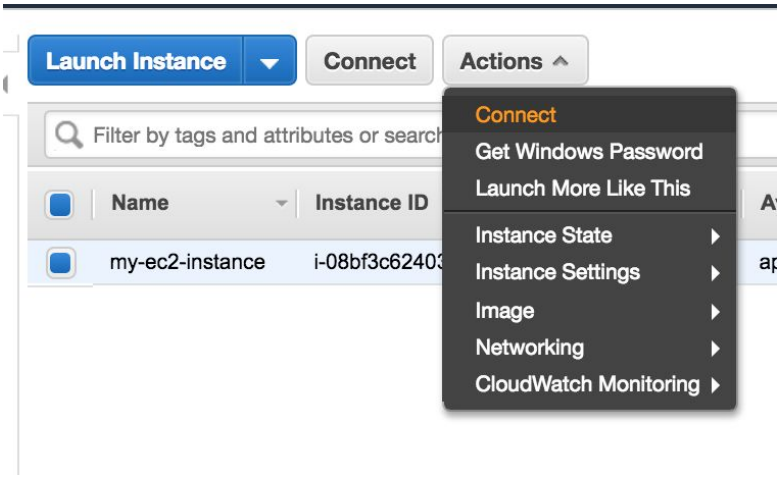
- Select “Choose file” and browse to the key file you downloaded earlier in this section.

- Select "Decrypt Password"

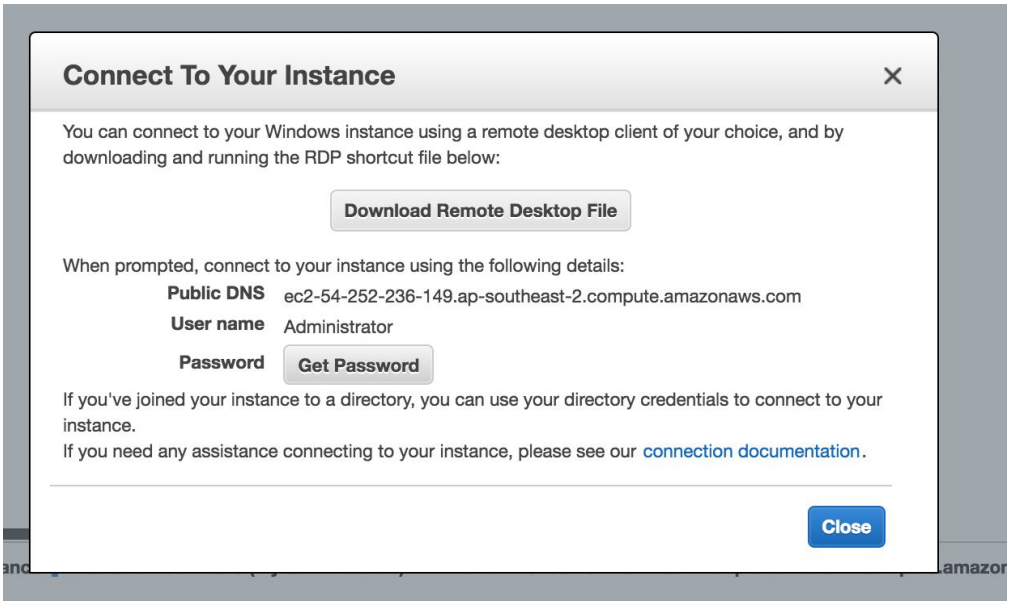


- Keep the details safe somewhere.

- Again with the instance selected, click "Actions" and select "Connect"



- Popup: "Connect To Your Instance" select "Download Remote Desktop File"



- This should download, automatically open, and configure a Remote Desktop Protocol client to connect to your instance.

If clicking the "Download Remote Desktop File" didn't work for you:
- Open your RDP client, enter the DNS value into the address field
- When prompted enter the username and password.

- Note: when you connect you may be prompted by security warnings. This is not a production environment, these warnings are ok.

- Back in the AWS console, let's take a look at how our server is doing.

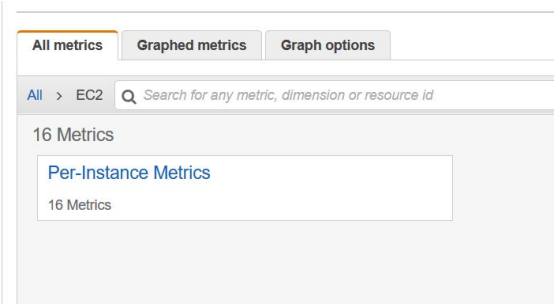
You might want to open this next link in a new tab, it helps to have different AWS services open in different tabs to save time when switching back and forth.

- Navigate to CloudWatch

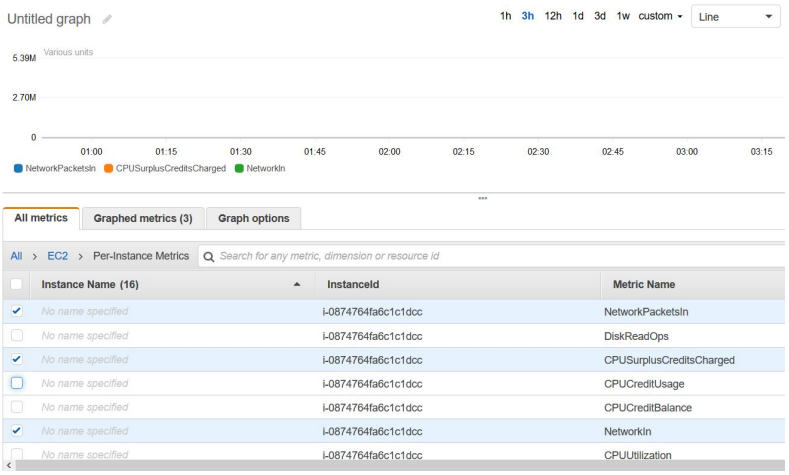
- CloudWatch is the central location of all monitoring and logging in AWS.

- Select "Browse Metrics"

- Towards the bottom, with the tab "All metrics" selected, select "EC2" then "Per-Instance Metrics"



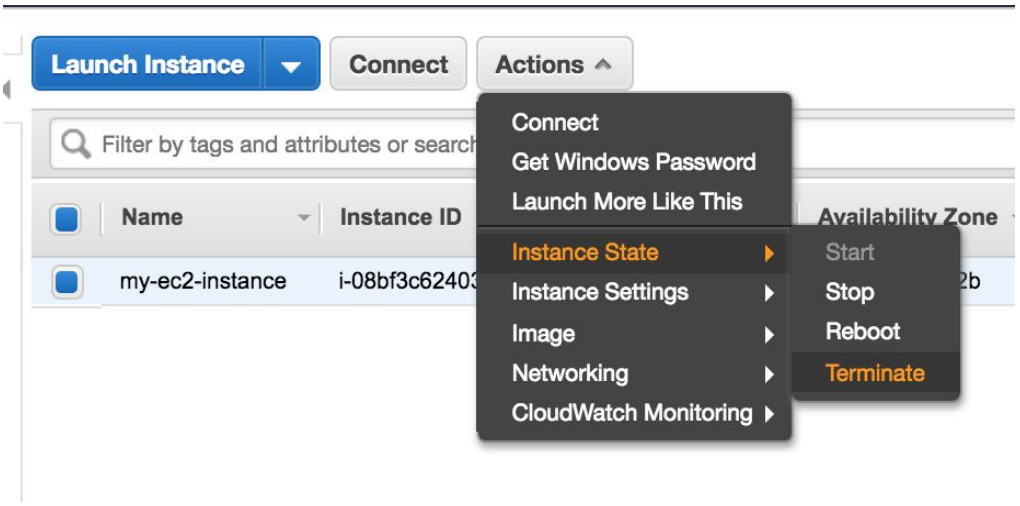
- Have a browse around the available metrics. Over time you will be able to see these metrics graphed here.



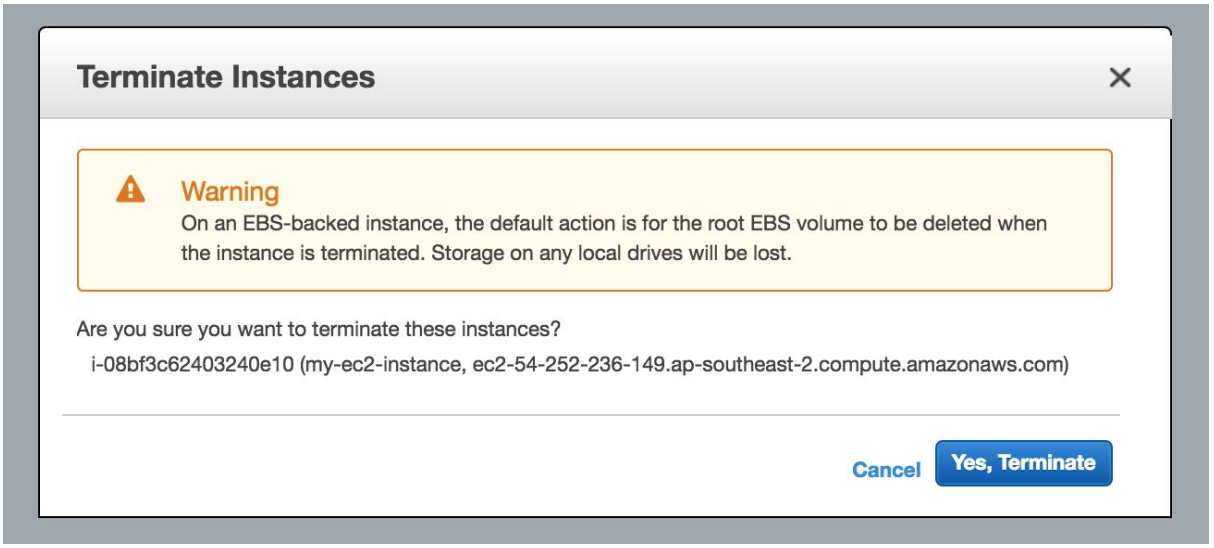
- Now let's clean up. We don't need this server anymore:

- Change your region back to Sydney.

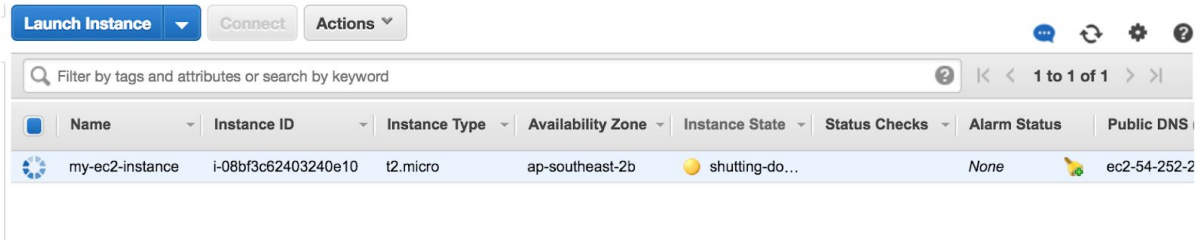
- Back in the EC2 console page, with the instance selected, select "Actions", "Instance State", "Terminate"



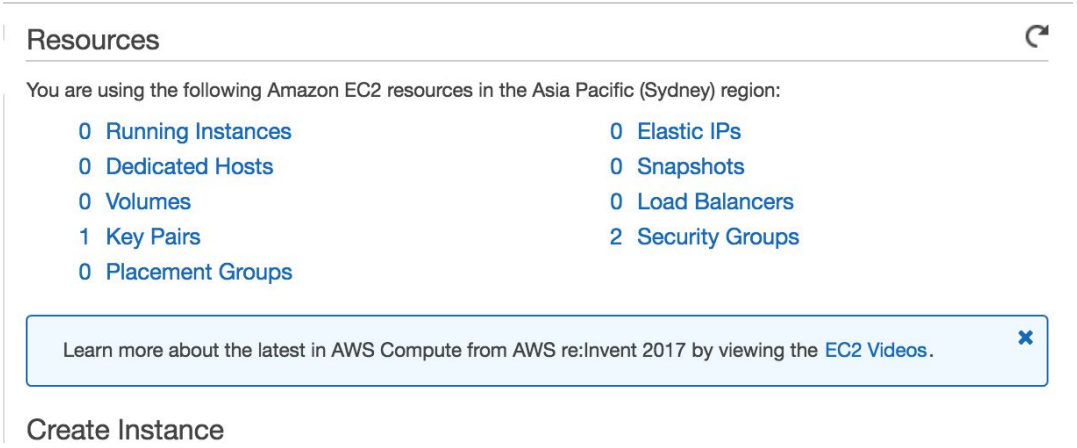
- Review the warning to make sure you are terminating the right instance, if so, select "Yes, Terminate"



- Note the Key Pair is still in the account (along with some other stuff), but that's ok we will use it later.



- Return to the "EC2 Dashboard", you should have 0 (Zero) running instances.



END

Session 2 - CloudFormation

In this session we will get introduced to AWS CloudFormation.

Take the following steps, one at a time. There is plenty of time to complete the session and ask any questions along the way.

Steps

- Lets create a CloudFormation template

- Open your favorite text editor (e.g. VSCode) and create a new YAML file.
- File new
- Save as "my-template.yaml"

Build a CloudFormation Template

- Now let's build the template

- First we add the template version:

```
AWSTemplateFormatVersion: 2010-09-09
```

- In this simple template we are only going to use the 'Resources' section:

- In the Resources section we will add an entry for each resource we are going to create. In this case that is one EC2 Instance

- (As we build the YAML file, look to ensure the tab (spaces) are correct)

```
Resources:
  myInstance:
    Type: "AWS::EC2::Instance"
```

- The "myInstance" element is used to refer to our instance elsewhere in the template.
That's not of much use in this template, but will be useful in more complex templates later.

- Notice that we have used an AWS specific Type of "AWS::EC2::Instance". This is where we set that we are creating an EC2 Instance

- Lets configure the instance with some properties. Create the following lines, read for some values:

```
Properties:
  ImageId:
  InstanceType:
  KeyName:
```

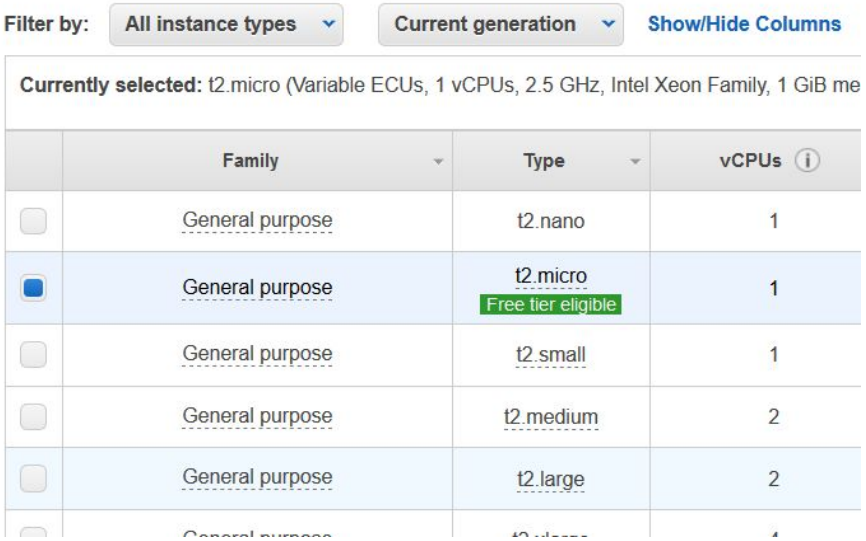
- Let's find the Image Id. When we created the image 'by hand' this was listed during Step 1.

- While we are creating this template, open a browser tab, in the EC2 Dashboard, and select "Launch instance"

- In "Step 1" look at the OS/Images, scroll to "Microsoft Windows Server 2016 Base"

- See the Amazon Machine Image ID (ami-xxxxxxx) written next to the AMI name.
- Make a note of this value.

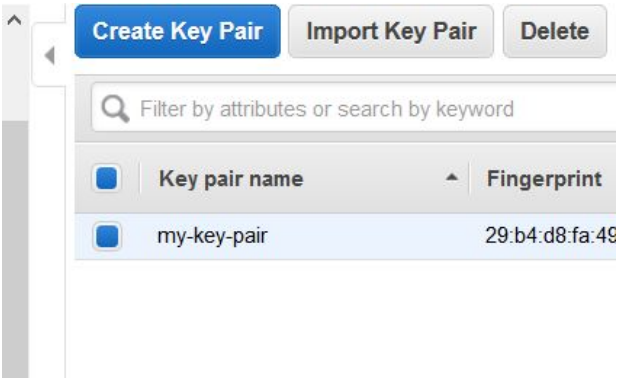
- Let's find the InstanceType.
- Select the AMI in the wizard to see "Step 2"



- In "Step 2" look at the Type we used. It was "t2.micro"
- Make a note of this value.

- Finally lets find the name of a Key Pair we can use.

- We created a Key Pair in the first session today. Let's find its name.
- Cancel the "Launch Instance" wizard.
- On the left hand side, scroll down to "Key Pairs"



- You should have one Key pair name in here. Make a note of this value. It should be the same name as the PEM file your downloaded in Session 1 (that's another way to find the name)

- Back in the CloudFormation template, enter the values we found into the relevant lines:

```
    Properties:
      ImageId: ami-82a458e0
      InstanceType: t2.micro
      KeyName: my-key-pair
```

- The complete file should now look like something like this:

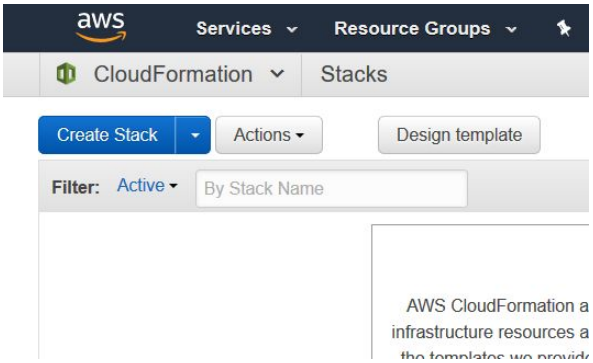
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  myInstance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: ami-82a458e0
      InstanceType: t2.micro
      KeyName: my-key-pair
```

- Save your YAML CloudFormation template somewhere on your computer.

Build a CloudFormation Stack

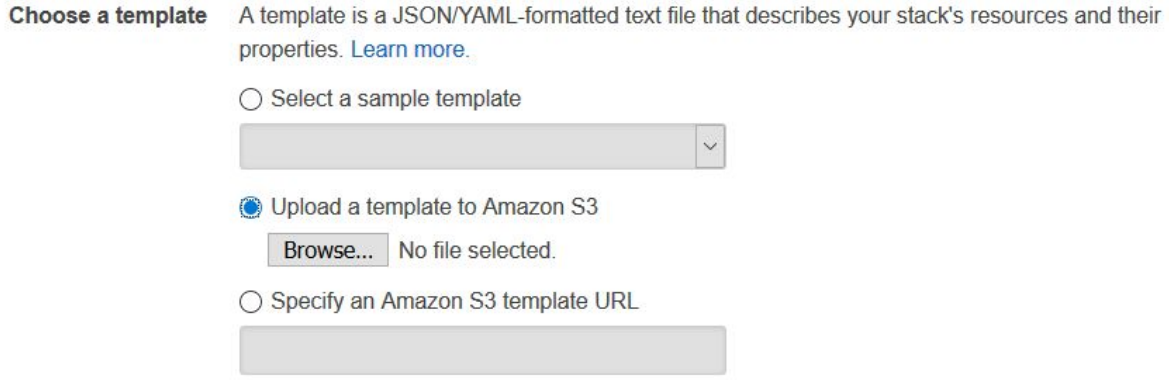
- Let's use the template to create a CloudFormation Stack

- Log into the AWS console
- Set the region to Sydney (stay in this region today, check from time to time)
- Navigate to CloudFormation



- Select "Create Stack"

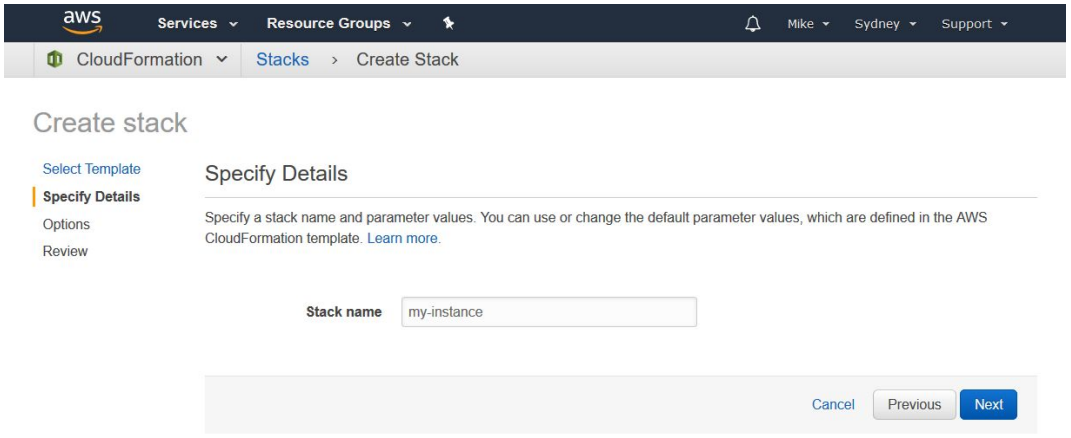
- From the next screen, select "Upload a template to Amazon S3" and click "Browse" and select the CloudFormation template you created earlier in this session.



- Select "Next"

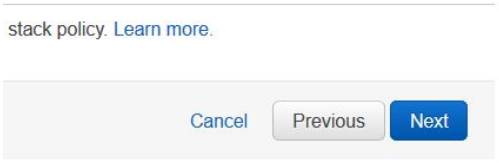
- If you are shown a validation error at this point, check that your YAML file looks like the example above. Are the spaces correct?

- Enter a name for your CloudFormation Stack. Enter "my-instance".

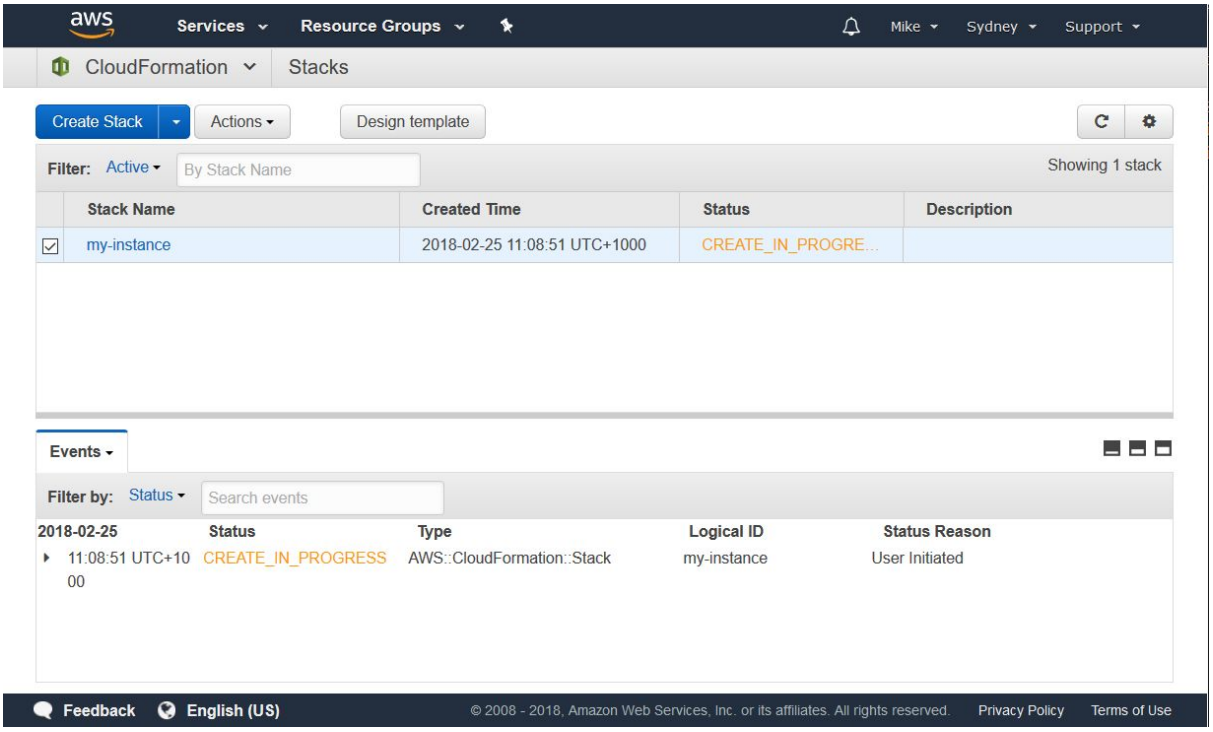


- Select "Next"

- The next screen provides some more advanced options for your stack. For now, scroll to the bottom of this page and select "Next"



- The next page shows a 'Review' or summary page. Scroll to the bottom of this page and select "Create"

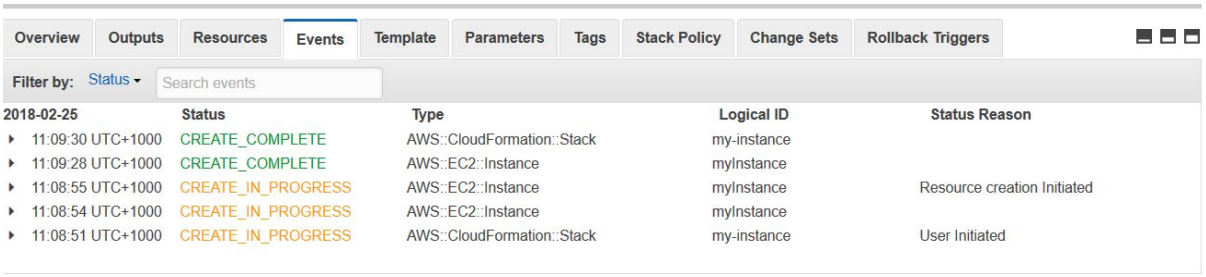


- You should now be taken to the CloudFormation stacks page. You can watch your stack being created.

- With your stack selected:

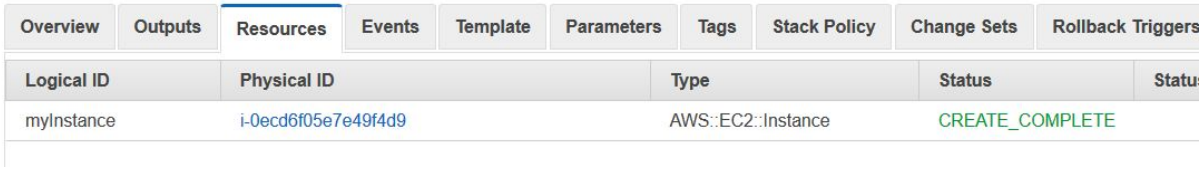
- Click the refresh button (within the page, NOT the main browser refresh button) from time to time, this will update all the information about the stack build.

- Review the "Events" tab, this shows you the steps being taken by the Cloudformation build engine.



- If there are any errors during the build, they will be shown here.

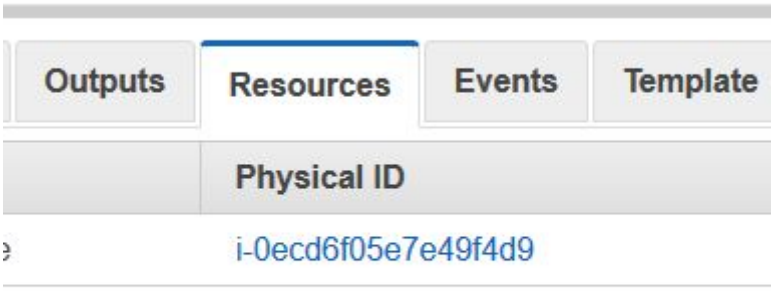
- Review the "Resources" tab, this shows the resources built in this stack. In this case you should see just one EC2 Instance.



- When the stack has built, with no errors, you should see the status as "CREATE_COMPLETE"

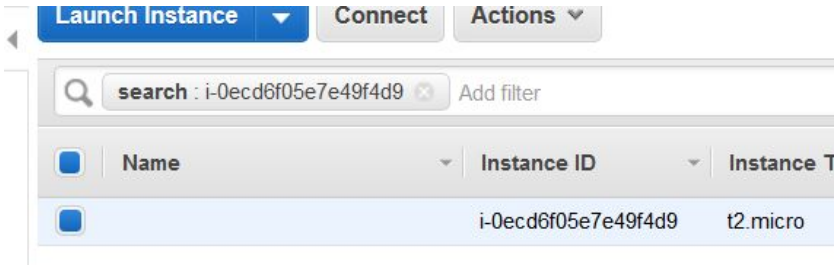
- Not seeing "CREATE_COMPLETE"? If the build has failed, see if you can see why from the "Events" tab. If you need help please ask :)

- With your stack selected, from the "Resources" tab click the "Physical ID" of the instance that the strack created.



- This will take you to the EC2 Instance list. Note the list of instances is now filtered to just the instance you created (which is your only instance)

- See that the Instance does not have a Name tag. Let's fix that... not in the console, but in the CloudFormation template.



Update a CloudFormation Stack

- Back in your text editor, with your CloudFormation YAML file loaded, add a Name tag to your instance. Under KeyName, and on the same level as KeyName, add:

```
      Tags:
        -
          Key: Name
          Value: my-ec2-instance
```

- Again make sure the spacing is correct, your template should now look something like this:

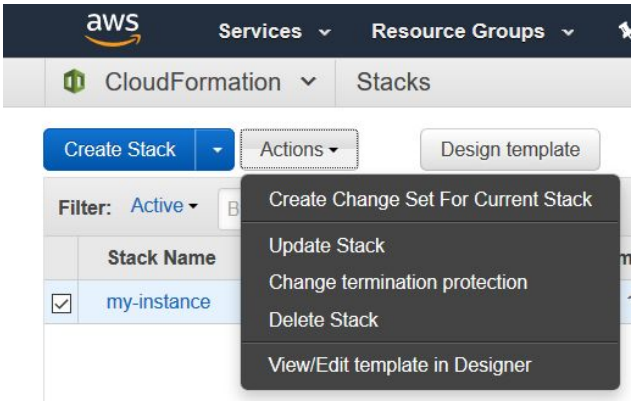
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  myInstance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: ami-82a458e0
      InstanceType: t2.micro
      KeyName: mykey
      Tags:
        -
          Key: Name
          Value: my-ec2-instance
```

- Save your updated template.

- Now, let's update the stack so the changes take effect.

- Navigate to CloudFormation

- With your stack selected, select "Actions", "Update Stack"

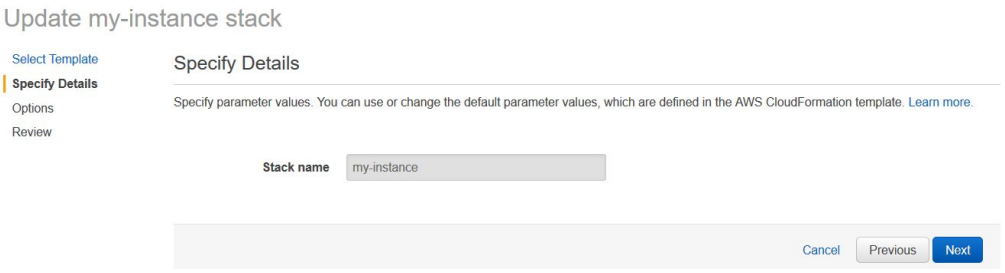


- From the next screen, select "Upload a template to Amazon S3" and click "Browse" and select your updated CloudFormation template.

- Select "Next"

- If you are shown a validation error at this point, check that your YAML file looks like the example above. Are the spaces correct?

- There should be nothing to change on the next screen, select "Next"



- The next screen provides some more advanced options for your stack. For now, scroll to the bottom of this page and select "Next"

- The next page shows a 'Review' or summary page.

- Notice the "Preview your changes" section, this gives you a very high level summary of what will change in the stack.

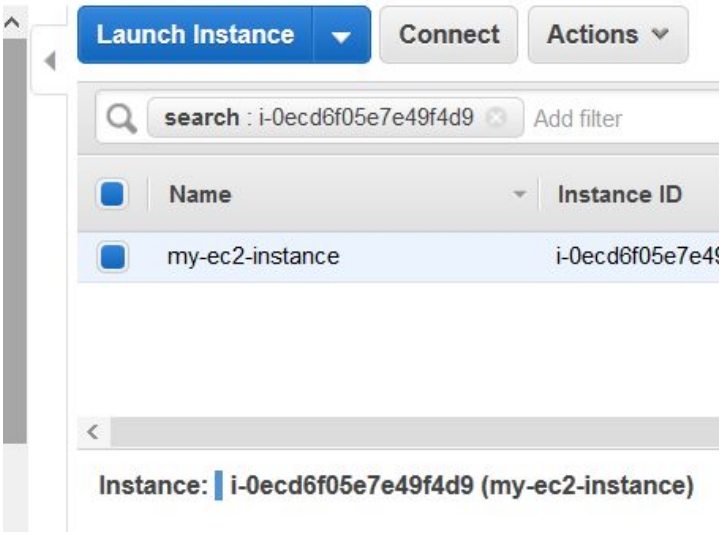
- Notice that while the EC2 Instance will be updated, it will not get replaced, so the instance will remain running.

- Select "Update"

- As before watch CloudFormation do its work, refresh, and review the "Events" tab.

- When the stack status shows as "UPDATE_COMPLETE", follow the link from the "Resources" tab to the instance view.

- Notice that the instance is now Named.



- If you have time you can connect to the server via RDP if you wish. When you're finished...

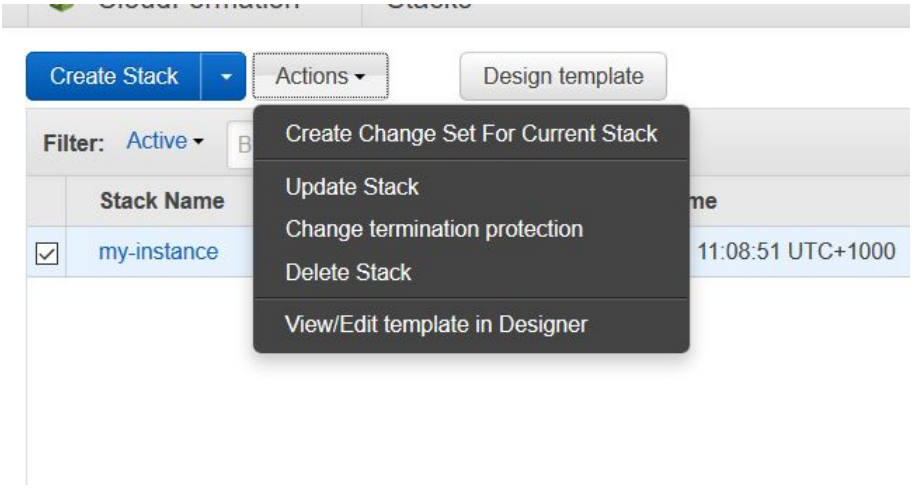
- We don't need this server anymore, let's clean up.

Delete a CloudFormation Stack

- Cleaning up the CloudFormation stack is super easy, and super clean. The instance and all related configurations (but not the Key Pair as this wasn't created in the stack) are gone in one action:

- Navigate to CloudFormation

- With your stack selected, select "Actions", "Delete Stack"



- Review the warning and, if you are ready select "Yes, Delete"

- Monitor as the stack is deleted.

- Want to build the server again, name tag and all? Just build a new stack with your saved template!

Extra Challenge

While we wait for everyone to complete this session….

- Can you make a change to the CloudFormation template to create 2 servers?
- Are the servers created one after each other, or do they build at the same time?

END

Session 3 - Virtual Private Cloud

In this session we will get introduced to AWS VPC.

Take the following steps, one at a time. There is plenty of time to complete the session and ask any questions along the way.

Steps

- Log into the AWS console
- Set the region to Sydney (stay in this region today, check from time to time)

- The nice thing about CloudFormation templates is that they can be easily shared around. In the resource file for this session you should find 'vpc-network.yaml'

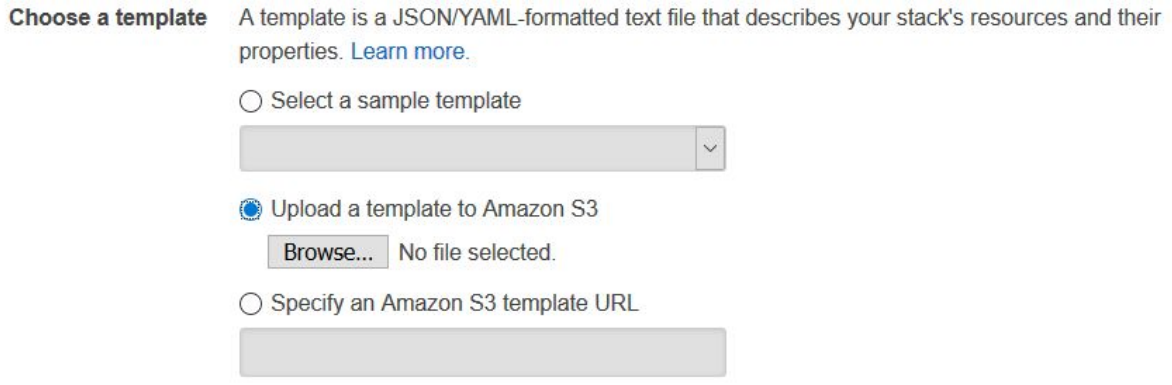
- Take a look at the file, we've added comments along the way to explain how it works. Notice that this template uses 'Parameters' to pass values in to the stack as it builds.

This template creates a VPC with four subnets. Two of these subnets are 'Public' in that they have access to and from the Internet. Two of the subnets are 'Private' in that they have no access to the Internet.

- Navigate to CloudFormation

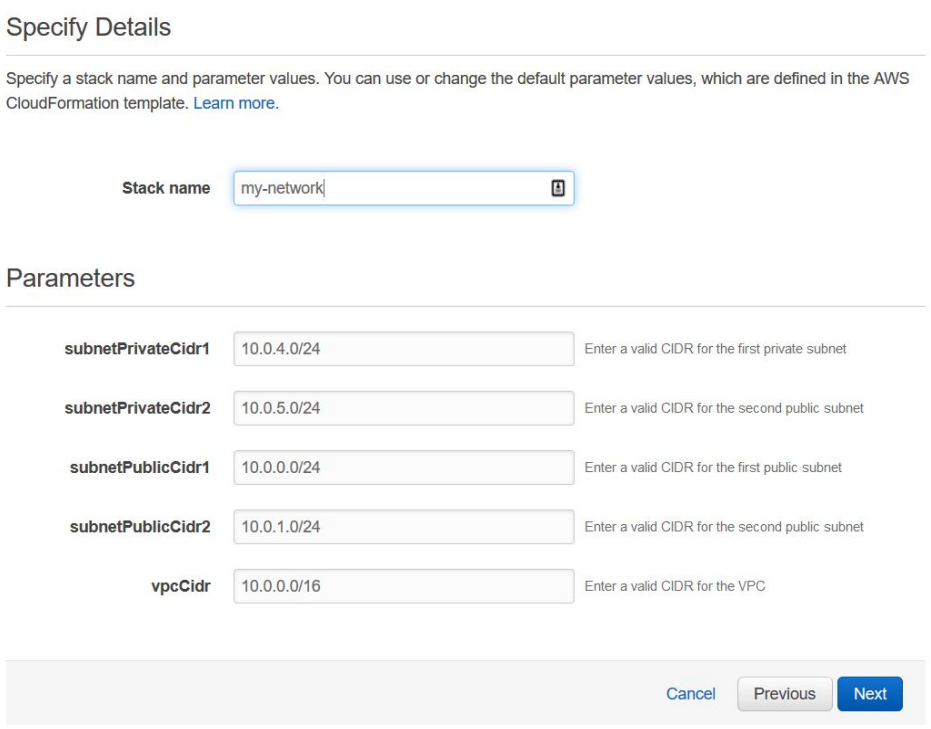
- Select "Create Stack"

- From the next screen, select "Upload a template to Amazon S3" and click "Browse" and select the CloudFormation template 'vpc-network.yaml'.



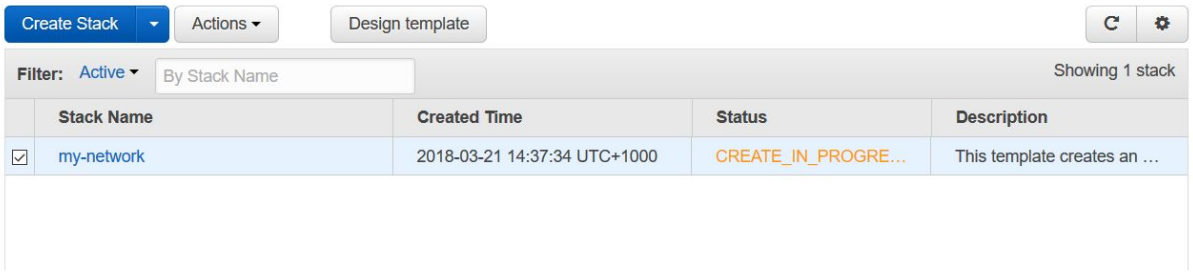
- Select "Next"

- This time as well as a stack name, you have the opportunity to alter some of the values that the stack will use.



- Enter a 'Stack name' and leave the other Parameters as default. Select 'Next'.

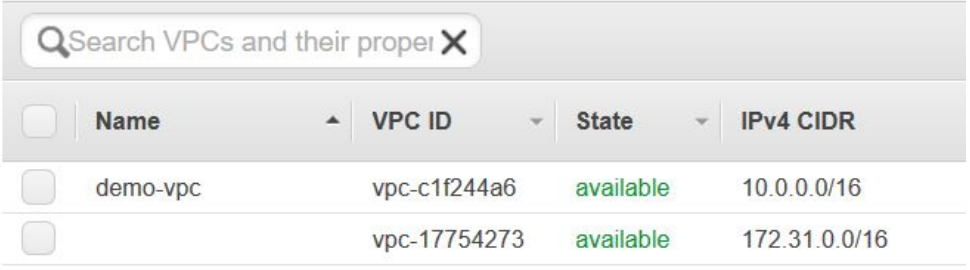
- Scroll to the bottom of the next page and select 'Next' and then select 'Create'.



- As before watch CloudFormation do its work, refresh, and review the "Events" tab.

- Navigate to VPC

- Take a look at the VPC you created.



- Notice that this is not the only VPC in your account. You should also see an (unnamed) 'default' VPC.

- We have now created a VPC with four subnets, two are public (in that it can route to and from the Internet) and two subnets are private.

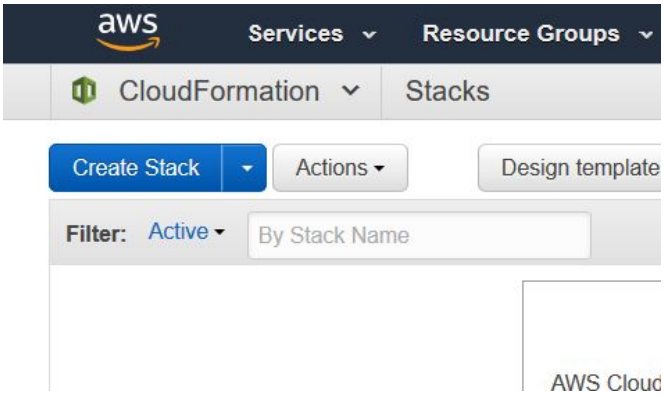
Testing the Virtual Private Cloud (VPC)

- Now let's create a couple of EC2 Instances to test with.

- First we will create an EC2 instance in a public subnet.

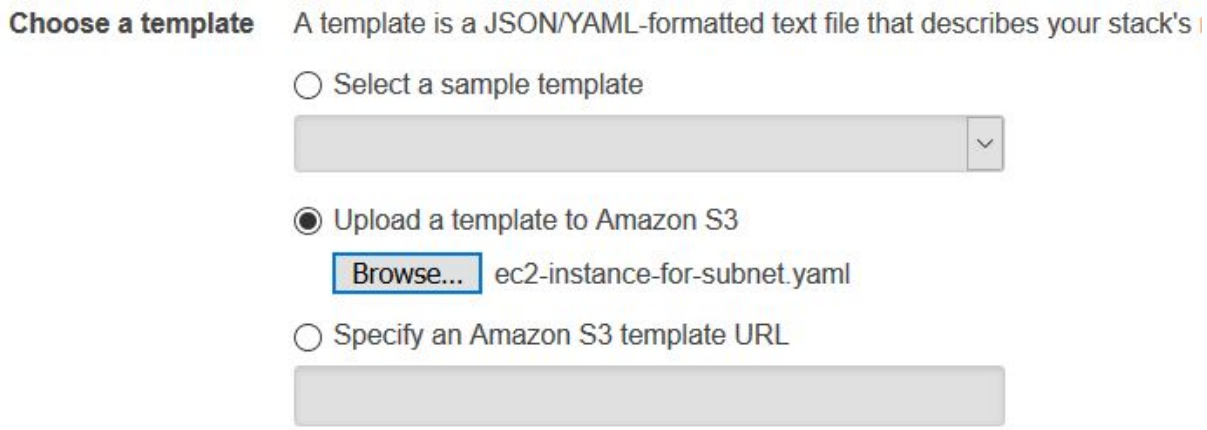
- Navigate to CloudFormation

- Select "Create Stack"



- Select "Upload a template to Amazon S3" and "Browse"...

- In the resources provided for this workshop, in the folder "Session 3" select the template: "ec2-instance-for-subnet.yaml"



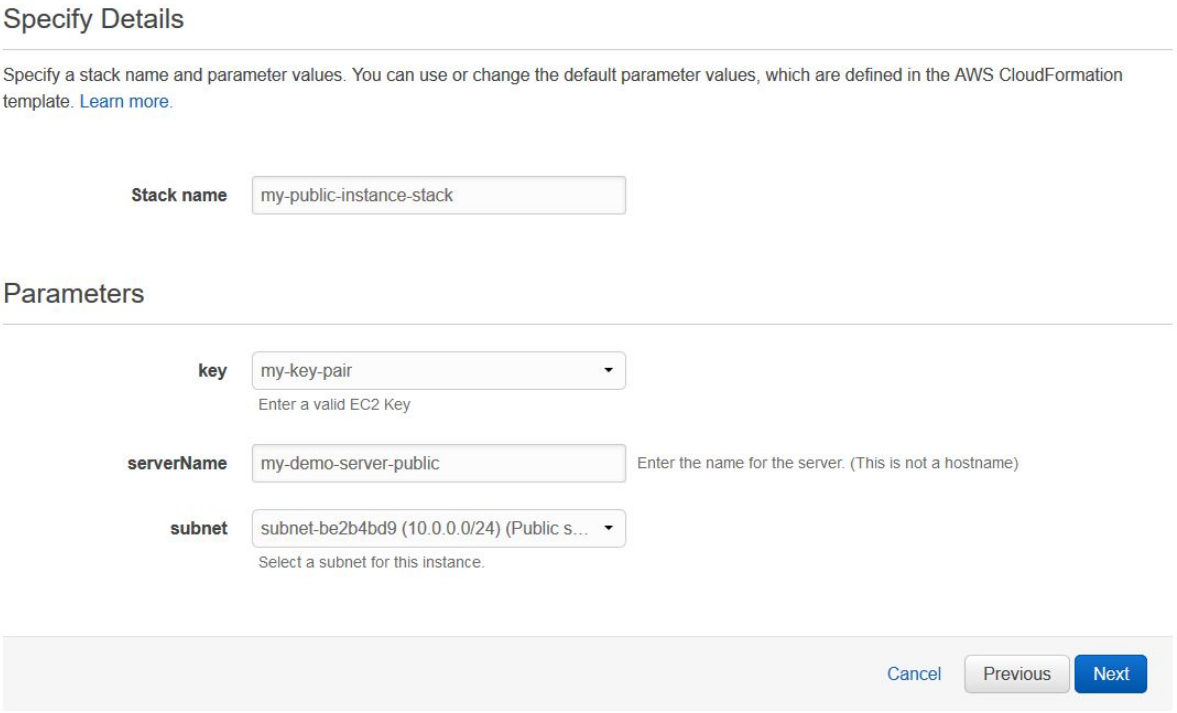
- Select "Next"

- This template has been created to prompt you for parameters before the stack is built.

- Enter a name for the stack "my-public-instance-stack" (this is not the name that will be given to the instance)

- The Key parameter is now a dropdown box. Select a key that you have on your machine.

- Enter a server name "my-demo-server-public"
- In the subnet dropdown menu you will have a few options. Select the subnet that has "(subnet-public-1)"
- Before moving on, make a note of the subnet ID and its corresponding name. We will need this information later. (e.g. subnet-public-1 : subnet-beb4bd9de9893478983)



- Select "Next"

- Leave all the settings on the next screen.

- Scroll to the bottom and select "Next"

- Scroll to the bottom of the Review page and select "Create"

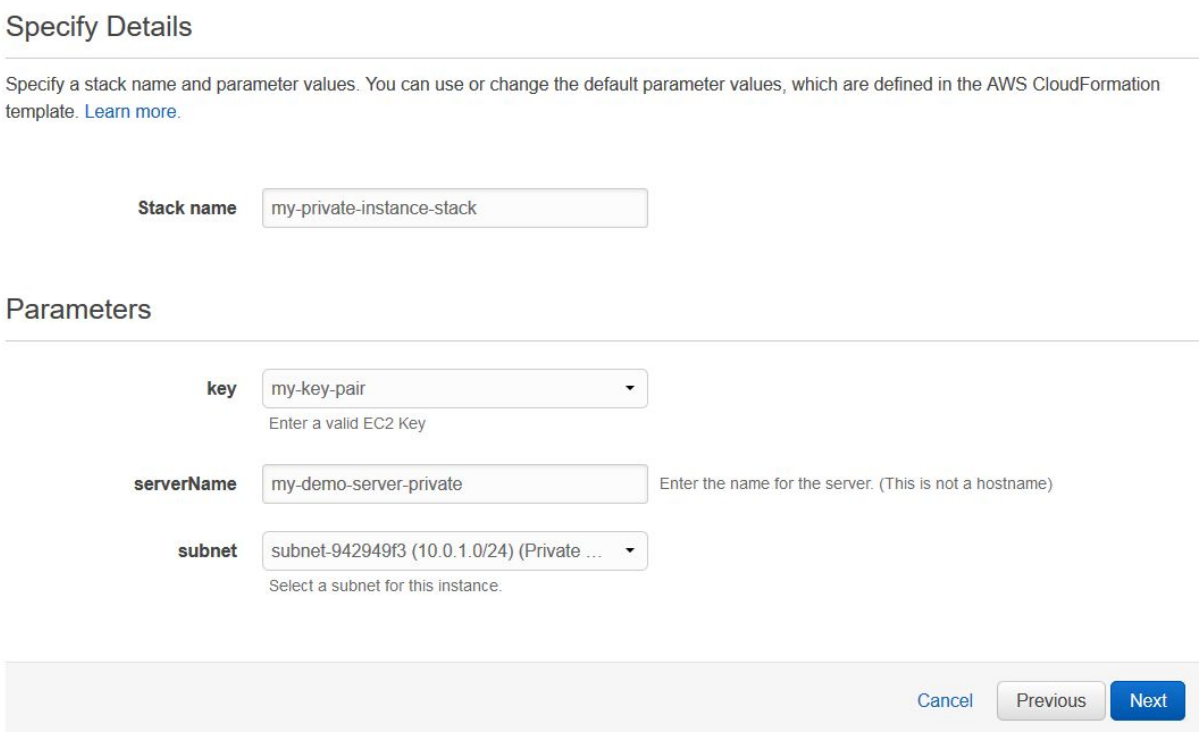
- We won't wait for this stack to create, let's create another EC2 Instance in the private subnet.

- Select "Create Stack" and follow the same process, but this time:

- Enter a name for the stack "my-private-instance-stack"

- Enter a server name "my-demo-server-private"

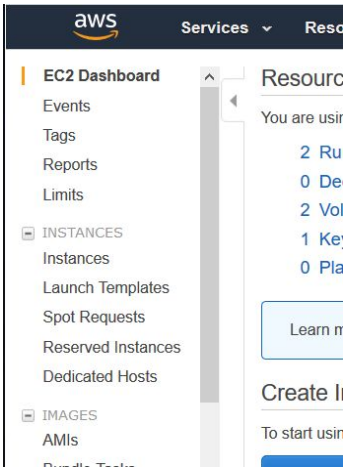
- Select the subnet with "(subnet-private-1)"



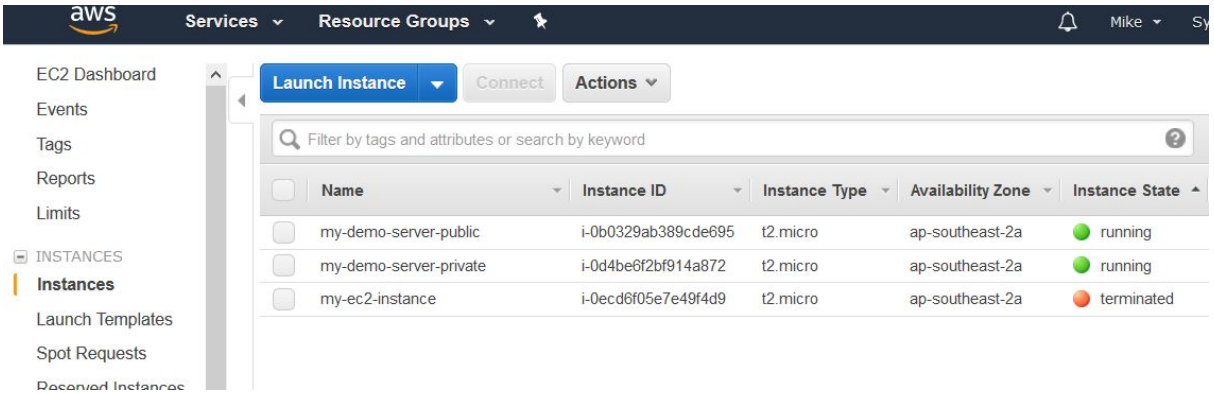
- Select “Next”, “Next”, then “Create”

- By now, our public EC2 Instance may have built. Let's connect into the server.

- Navigate to EC2
- Select "Instances" on the left hand side.



- By now, you should see two running EC2 instances, if you need to expand the Name column.



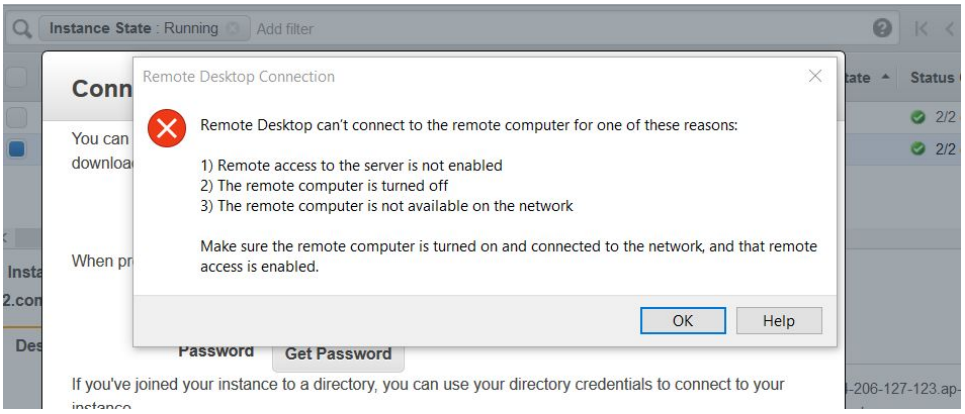
- Connect with RDP into the public server:

- Select the public instance, "Actions" and "Get Windows Password"

- Password not available yet? Hold on a minute, and try again.

- Try to connect to the public instance via your RDP client...

- Its not working!!



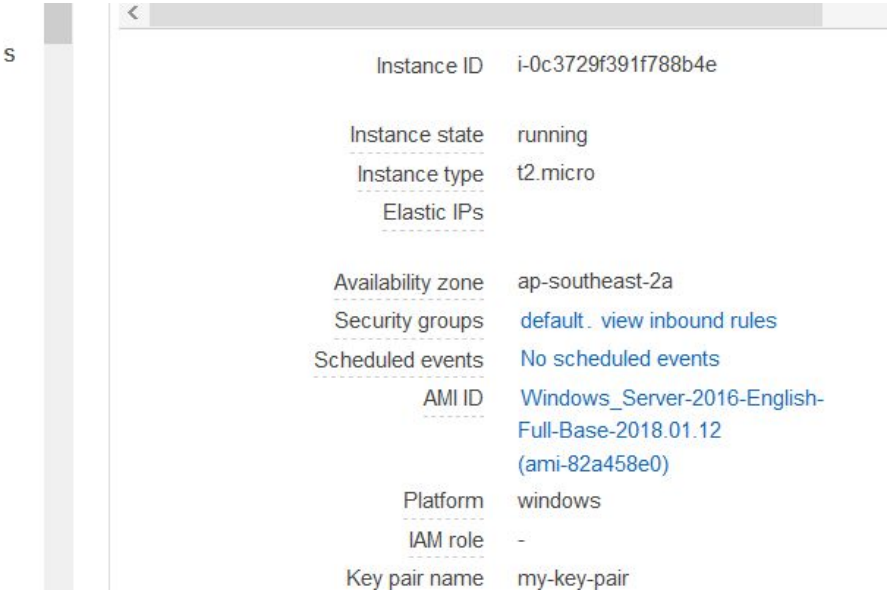
- In our VPC we have Security Groups that control the traffic that can reach our Instances.

- Both of our instances are in the 'default' security group.

- Let's create a rule in the default security group to allow access to our Instances via RDP.

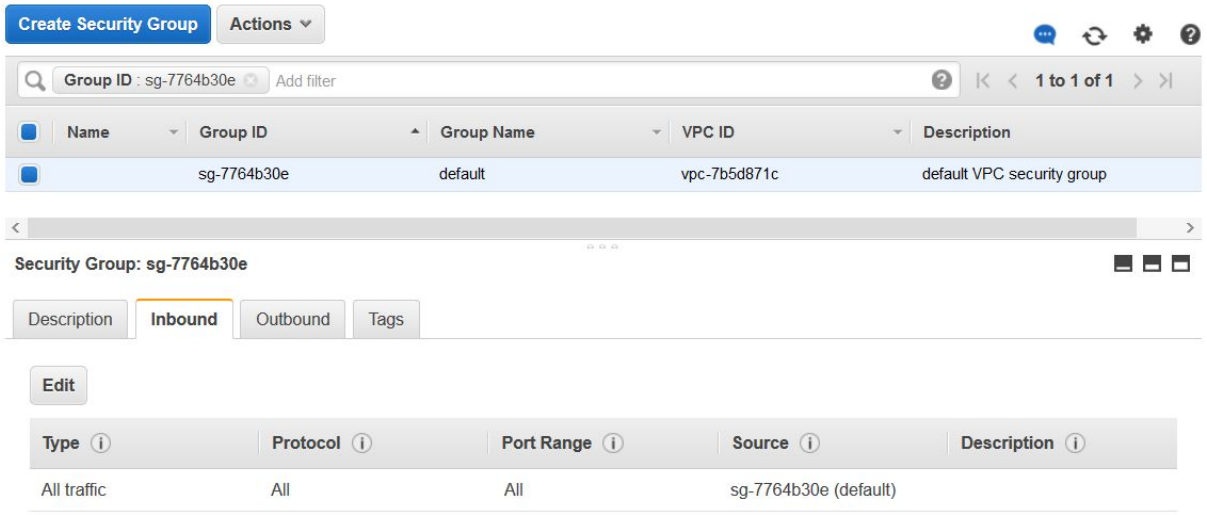
- From the EC2 list select either of the instances, look at the details section in the lower half of the screen.

- In the "Description" tab, find the "Security groups" and click on "default"



- On the following screen in the lower half, you will see the details of the security group.

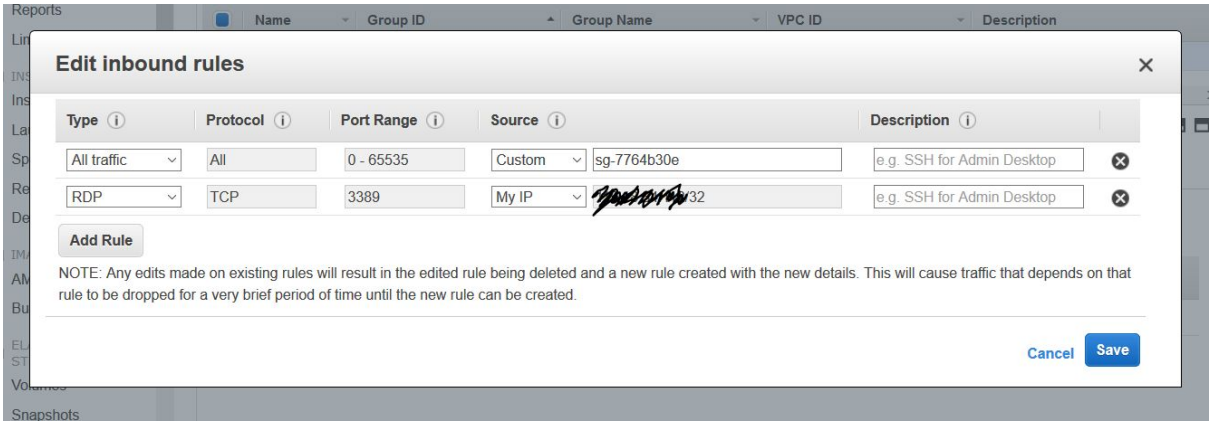
- Select the "Inbound" tab.



- Notice that currently, "All traffic" is allowed inbound to our instances, but the source is this (the same) security group.

- This means that the instances can RDP to each other, but we can't RDP from outside the network.

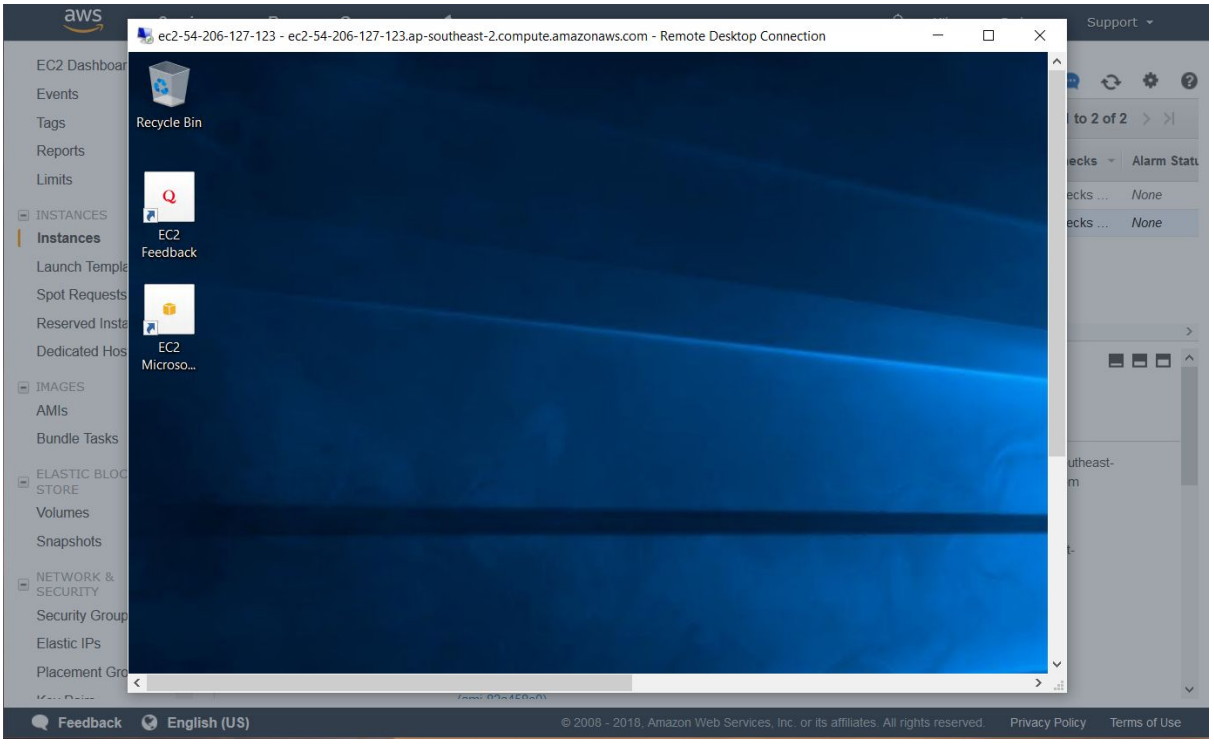
- Select "Edit", then "Add Rule"



- On the new rule, for Type select "RDP", and for Source select "My IP"

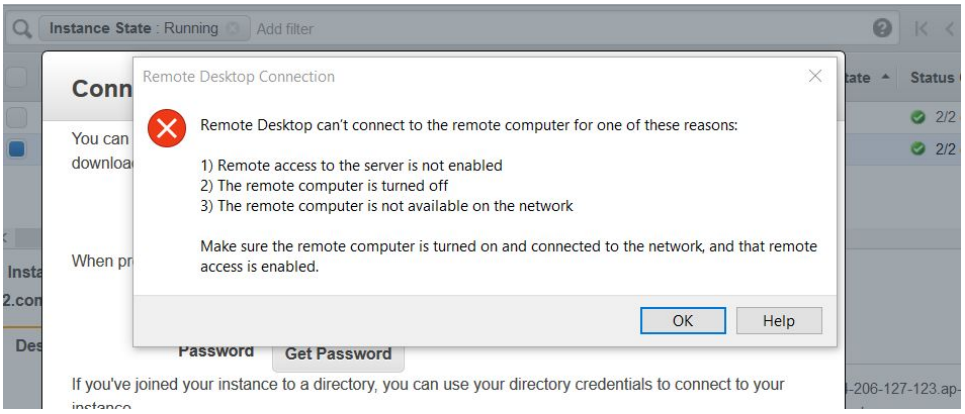
- By selecting "My IP" we restrict RDP access from the outside world to only machines using your Internet connection.

- Now, try again to RDP to your public instance.



- Yay, it works.

- OK, minimise the RDP session to the public instance. Let's see if we can RDP to the private instance:



- Of course, it doesn't work. That's because, even though the instance has been allocated a public IP address, its not 'visible' the the Internet through the VPC.

- But we can still get there:

- From the EC2 Instance list, select the private server, and in the details in the bottom half of the screen find "Private IPs".

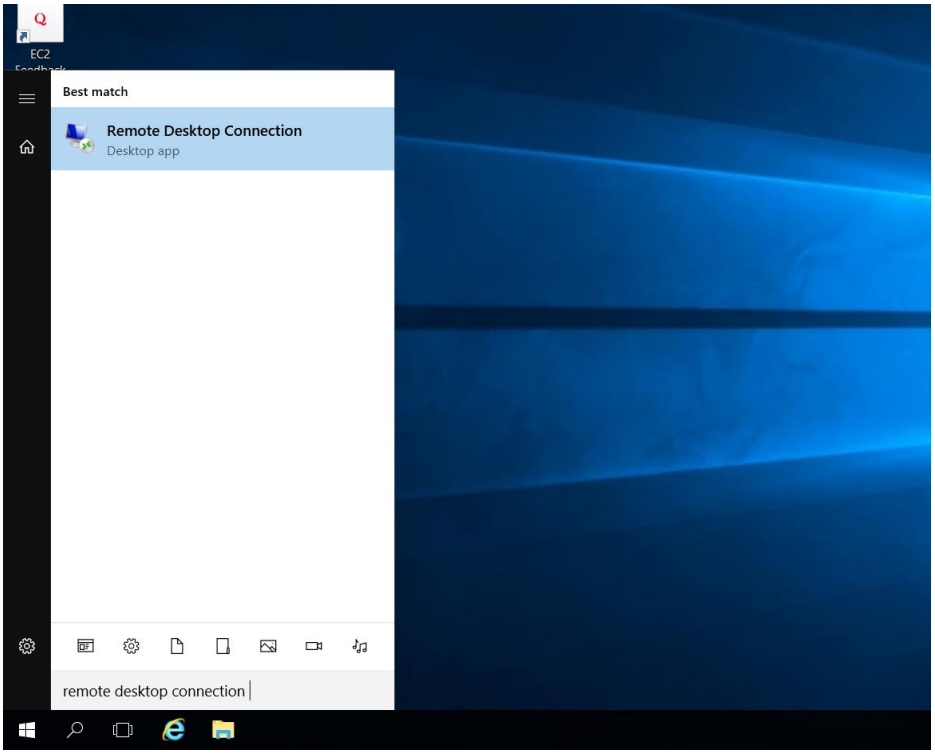
Public DNS (IPv4)	ec2-13-211-29-84.ap-southeast-2.compute.amazonaws.com
IPv4 Public IP	13.211.29.84
IPv6 IPs	-
Private DNS	ip-10-0-1-7.ap-southeast-2.compute.internal
Private IPs	10.0.1.7
Secondary private IPs	
VPC ID	vpc-7b5d871c
Subnet ID	subnet-942949f3

- Make a note of this address.

- Also take a moment to discover the password for this server.

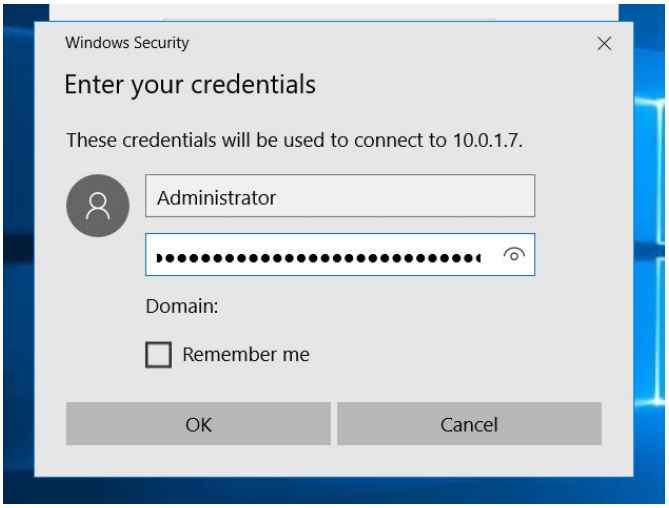
- Go back to the RDP session with your 'public' instance.

- From the 'Start' menu select "Remote Desktop Connection"

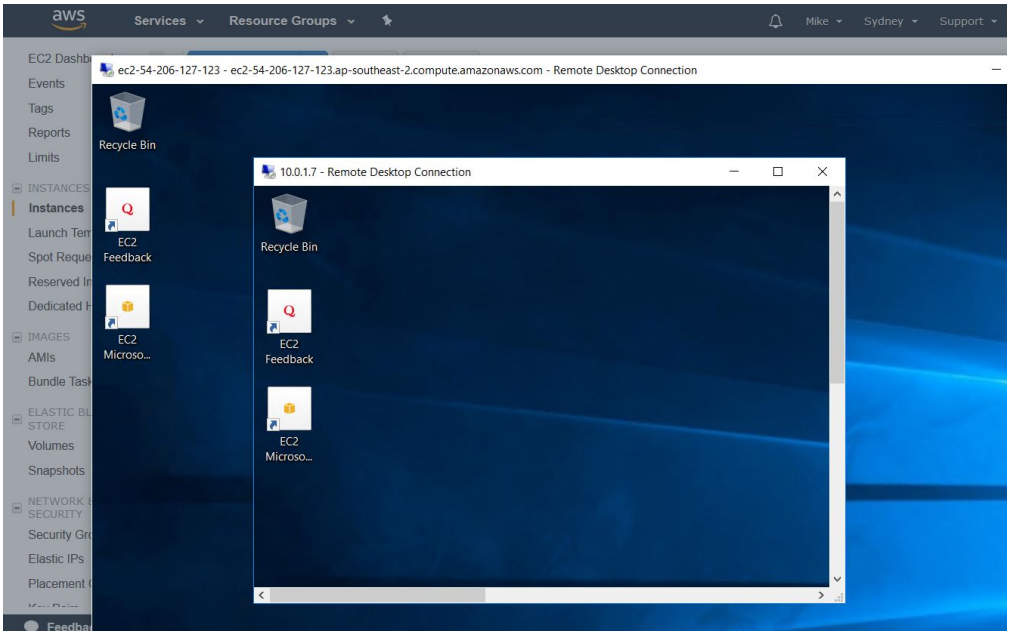


- Enter in to the "Computer" field the Private IP address of the private server that you noted down just before.

- Select "Connect"



- In the credentials box, enter the User name "Administrator" and the password for the private server.



- Yay, it works.

- You have created a private server, in a private network, that is accessible via what we call a bastion host.

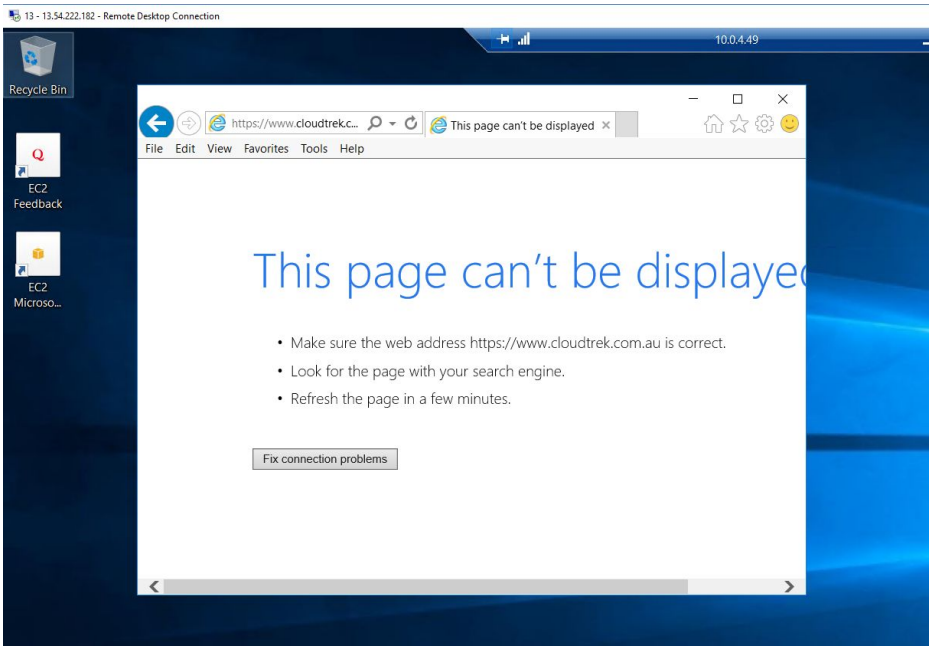
Before we finish…

- Let's look at Internet FROM connectivity for our EC2 Instances. It's often the case that servers need to connect to the Internet to download patches etc.

- From the RDP session with the public server, open up Internet Explorer and navigate to http://wikipedia.com (or any website).

- As this is a default 'super paranoid' server configuration, there will be a slew of warning messages, but you should end up looking at a webpage.

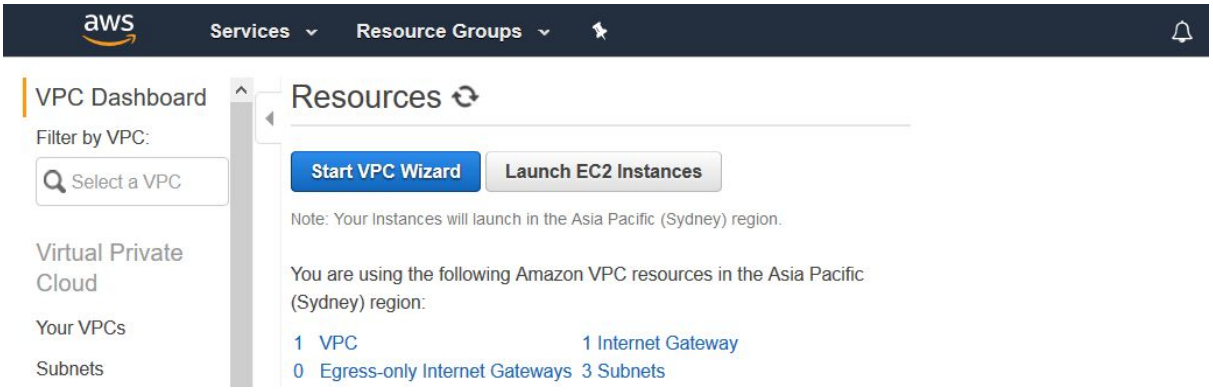
- If you do the same from the Private server… It does not work!



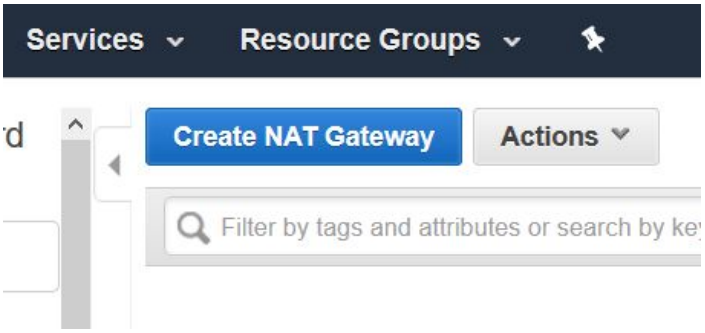
- Let's fix this. Let's allow our private server to have access to reach out to the Internet, keeping it in it's private subnet so that nobody can connect to it from the outside world directly.

- Leave your RDP sessions open….

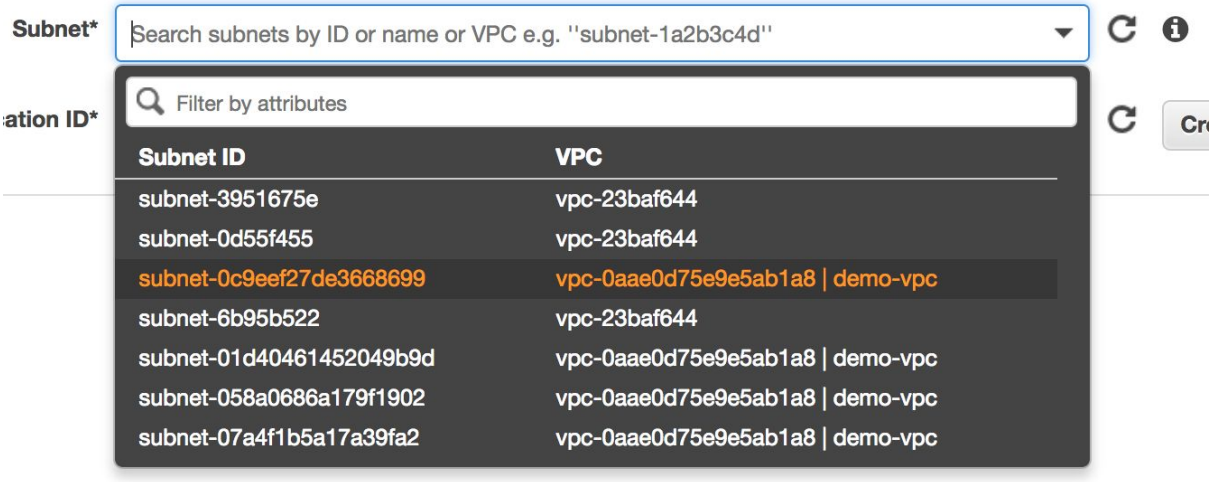
- Navigate to VPC in the AWS console.



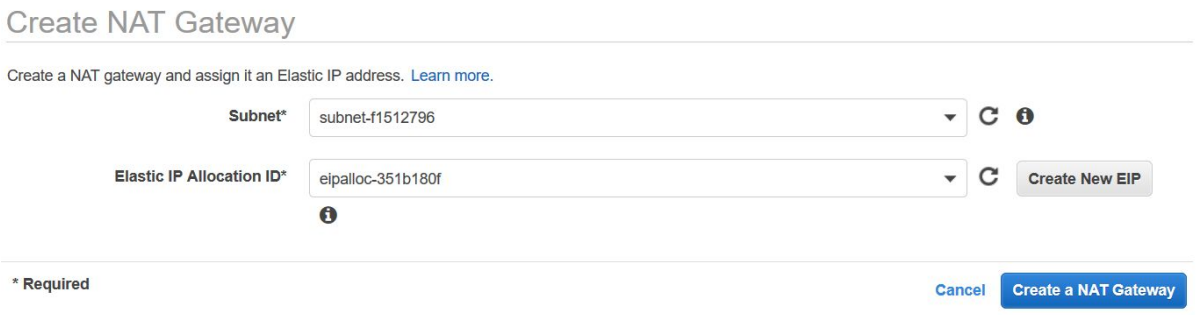
- From the menu on the left select 'NAT Gateways' then 'Create NAT Gateway'.



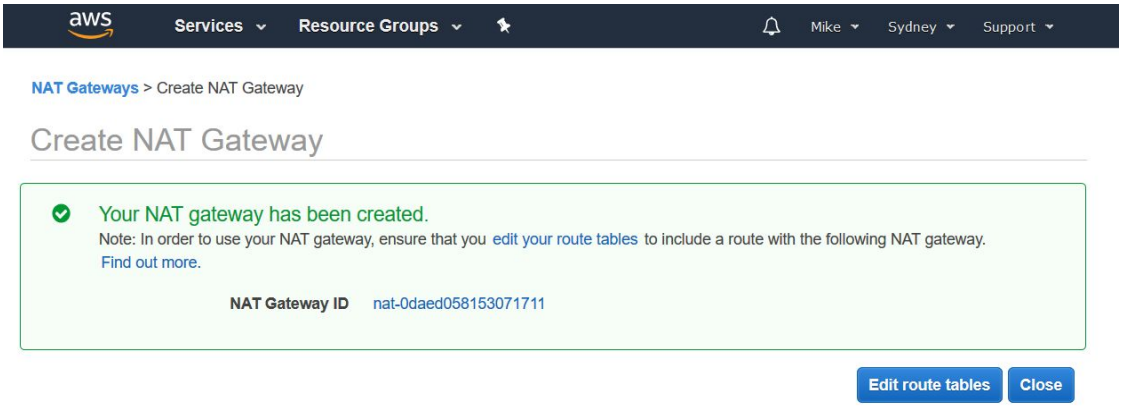
- For 'Subnet' choose the subnet ID that correspond to 'subnet-public-1' in demo-vpc. You should have this information noted down from earlier. If not, you will have to go to VPC -> Subnet and copy down the subnet-public-1 subnet ID.



- For 'Elastic IP Allocation ID' click on Create New EIP to fill in the box.

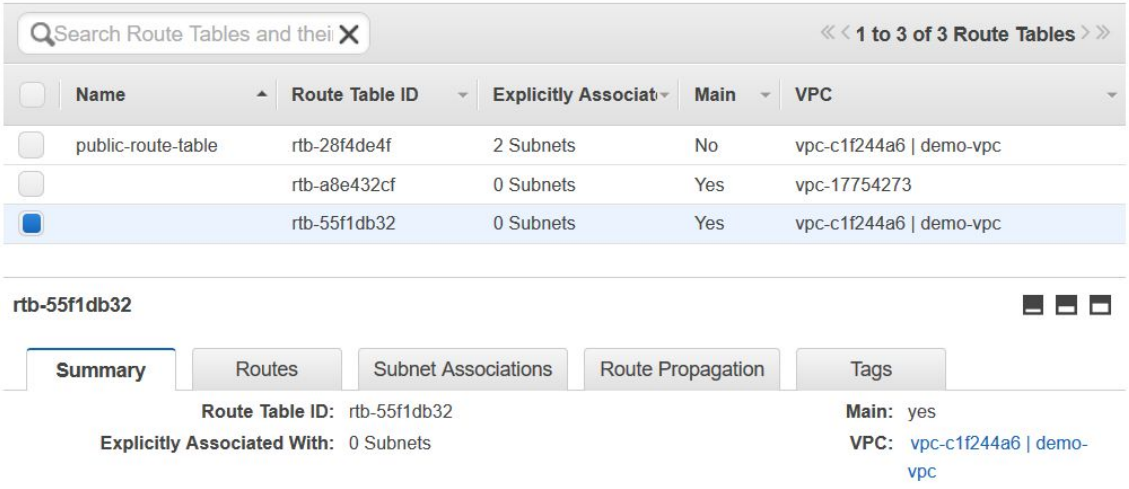


- Select 'Create a NAT Gateway'

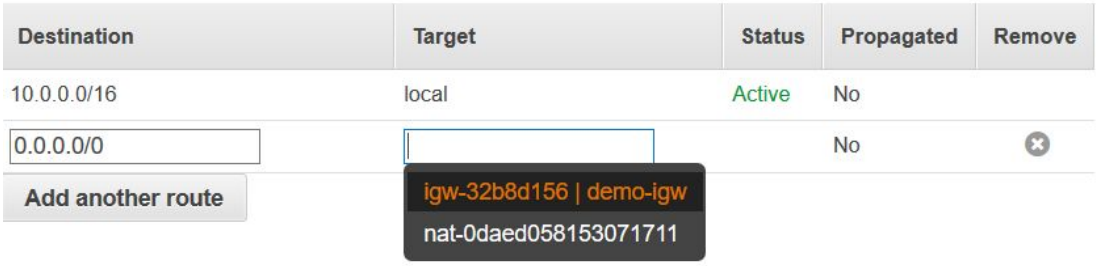


- Select 'Edit route tables'

- Look for the entry in this table for a 'Main' table for the 'demo-vpc'



- Select the 'Routes' tab, then 'Edit', then 'Add another route'



- In ‘Destination’ enter ‘0.0.0.0/0’ and under ‘Target’ start to type ‘nat..’ and select the ID of the NAT gateway.

- Select ‘Save’

- Go back to the RDP session on your Private server (inside the session to your public server. Try the web page again… it works!

- Note, you still can’t RDP directly to the Private server.

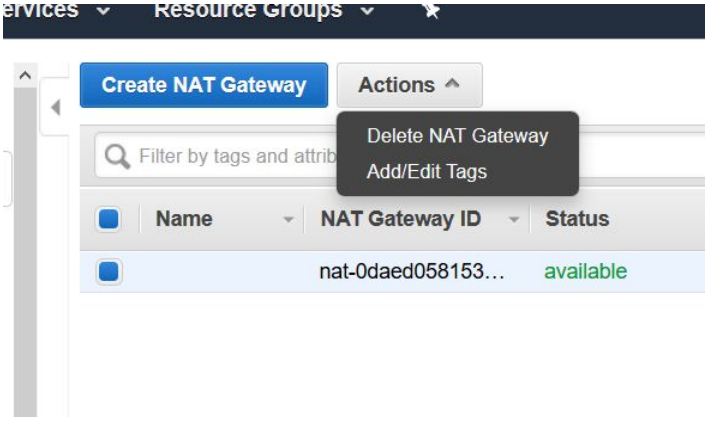
- Once your done experimenting with this environment its time to clean up.

Deleting your Resources

- Navigate to VPC

- Select ‘Nat Gateways’ from the left hand menu. If the gateway list is empty, you may have to click the ‘refresh’ button.

- With the Gateway selected, use the action menu and select ‘Delete NAT Gateway’ and then confirm again ‘Delete NAT Gateway’.



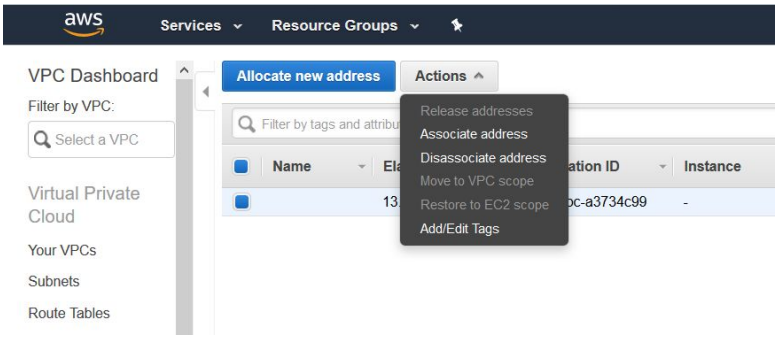
- Navigate back to CloudFormation

- Delete the Server stacks by selecting the stack, and selecting "Actions", "Delete Stack", then "Yes, Delete"

- Wait for these stacks to fully delete before doing the same to the network stack.

- One last thing, select "Elastic IPs" from the menu on the left.

- Select the Elastic IP entry, and select "Actions", "Release addresses", then "Release"



Extra Challenge

While we wait for everyone to complete this session….

- At the end of this session the Private server had access to the Internet.
- What are some of the reasons that this server needed access to the Internet?
    - Hint ‘AWS’
- Are there other methods for this server to get access to the Internet? Ways that are more controlled?

END

Session 4 - Bootstrapping

In this session we will get introduced to the concept of “Bootstrapping”.

Take the following steps, one at a time. There is plenty of time to complete the session and ask any questions along the way.

Steps

- In this session we are going to get a Windows server to do a “real job”. But we're not going to configure it by hand.

- Log into the AWS console

- Set the region to Sydney (stay in this region today, check from time to time)

- Navigate to EC2

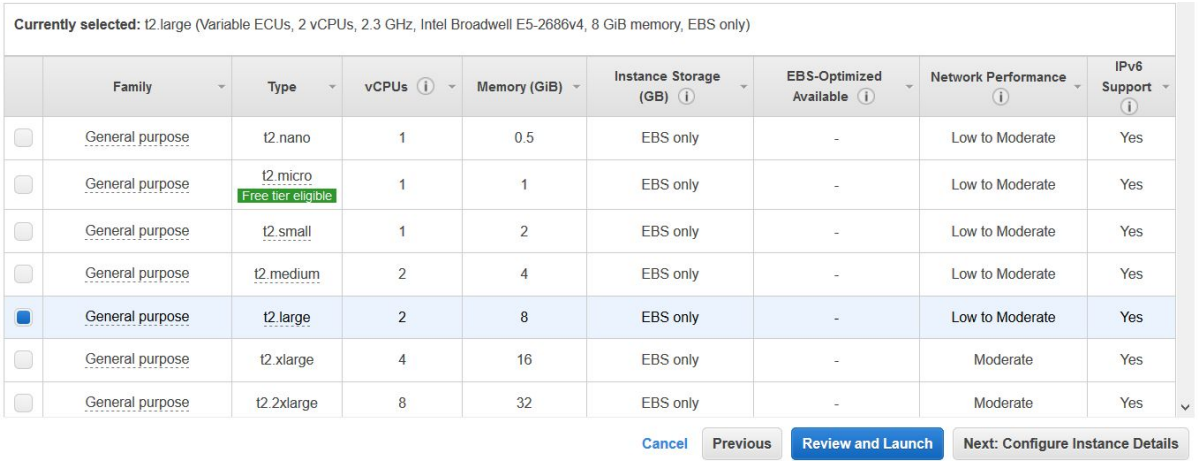
- First let's use the AWS console to build a web server:

Bootstrap an Instance with the Console

- Select "Launch Instance"

- Select "Microsoft Windows Server 2016 Base"

- This time, as we will be getting the server to do some work, and we don't want to wait too long, let's select a bigger instance type:
- Select "t2.large"

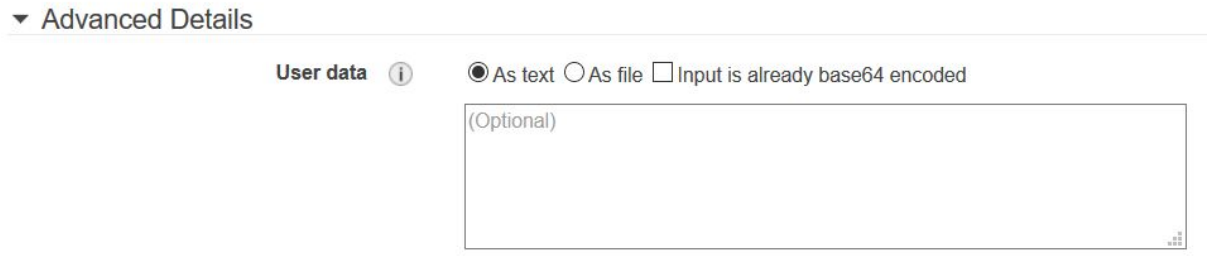


- Select "Next: Configure Instance Details"

- Take a look at the different settings options here.

- We will leave all the default values, but we will scroll down and expand the "Advanced Details":

- Here we see the "User data" entry.

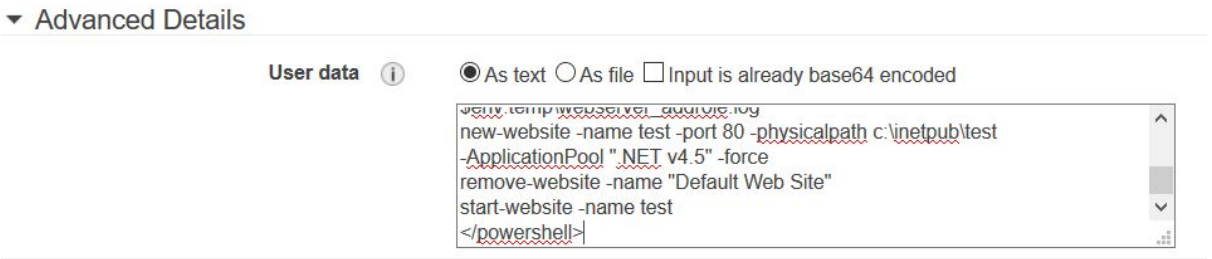


- Text we add to "User data" will be passed to the Instance as it boots, and if it's script, will be executed by an Administrator user automatically.

- In the resources for this session you will find user-data.txt

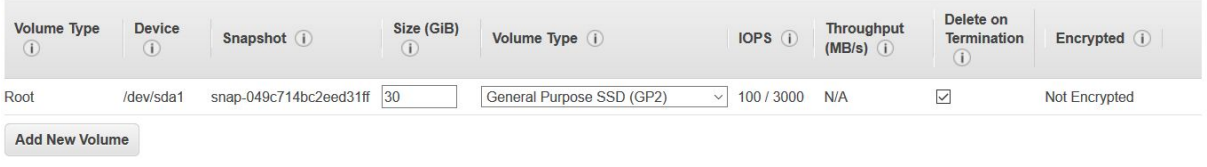
- Open up this file in a text editor and take a look. You will see that its some Powershell script that copies an ASPX web page, enables the IIS Web Role, and starts a website.

- Copy the text (completely) and paste it into the "User data" section of the EC2 console.



- Select "Next: Add Storage"

- We won't make any changes here, but see how we could change the size and number of disks that the server has.

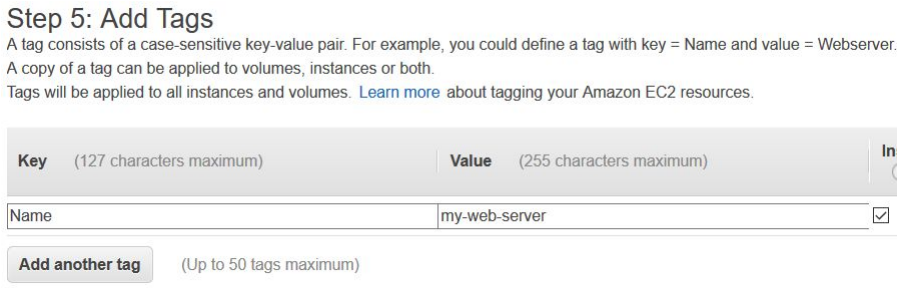


- Select "Next: Add Tags"
- We want the server configuration to be named

- Select "Add Tag"

- Enter "Name" for the key (note ensure the N is capital)

- Enter "my-web-server" for the value.



- Select "Next: Configure Security Group"

- Let's change the name of the Security Group so we can find it in the future, change the name to "web-server-group"

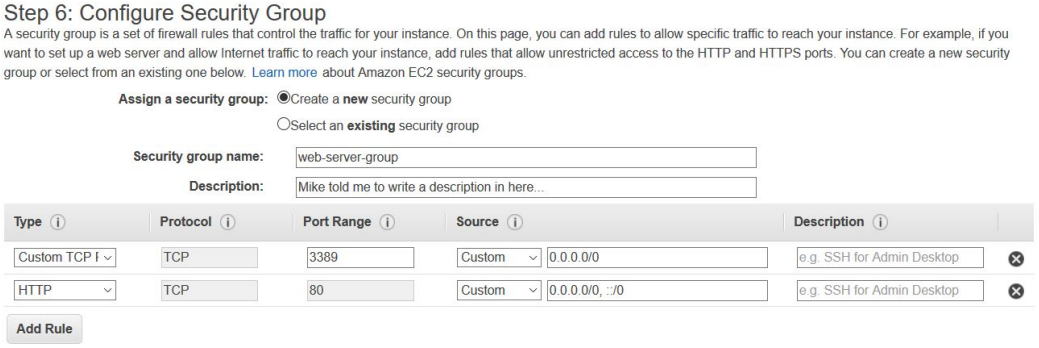
- Using descriptions is always a good idea in AWS, add a description so that when you look at this in the future you understand what it is

- Now we are going to allow connection on port 80 (HTTP) from the outside world

- Select "Add Rule"

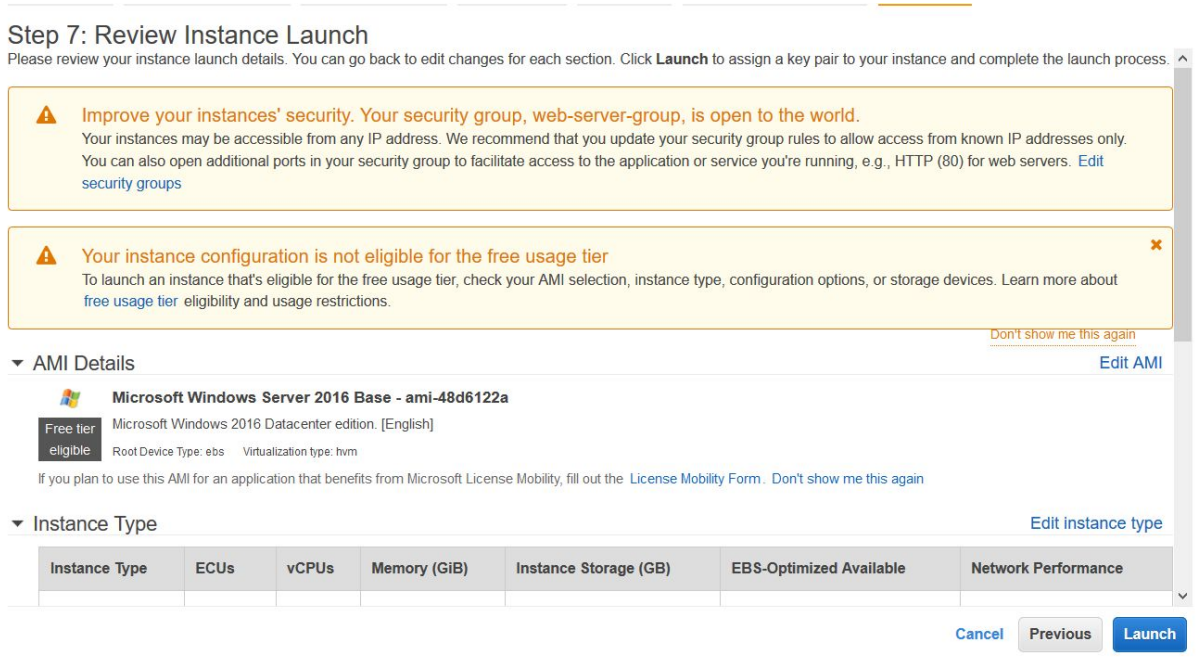
- In the Type dropdown select "HTTP"

- In the "Source" dropdown select "Anywhere"



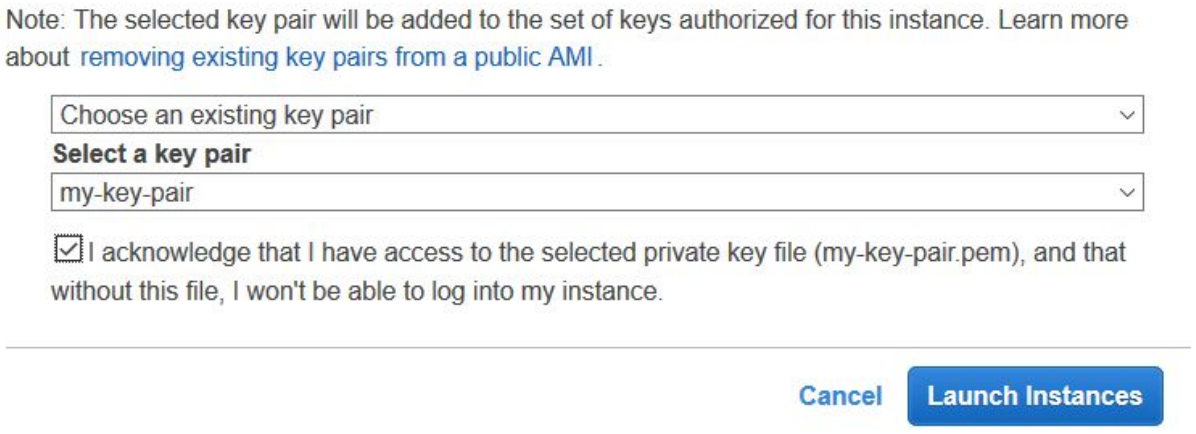
- Select: “Review and Launch”

- Review the details of the instance, and if all looks ok select "Launch"



- Select the Key Pair that you created earlier today

- You will need to acknowledge that you DO have the Key Pair



- Select: "Launch Instances"

- Scroll to the bottom of the screen and select "View Instances"

Bootstrap an Instance with CloudFormation

- Wow that was a lot of steps to create a server, there must be a better way?

- Well there are a few, but let's do the same thing again with CloudFormation.

- This time we have created the template for you, let's review it, you will find it in this sessions resources called "web-server-instance.yaml"

```
AWSTemplateFormatVersion: 2010-09-09

Description: This template creates an EC2 web server instance and allows you
to select the subnet it will use.  To keep it simple it only works in Sydney
(ap-southeast-2).

Parameters:
 serverName:
   Type: String
   Default: my-demo-server
   Description: Enter the name for the server. (This is not a hostname)

 subnet:
   Type: AWS::EC2::Subnet::Id
   Description: Select a subnet for this instance.

 key:
   Type: AWS::EC2::KeyPair::KeyName
   Description: Enter a valid EC2 Key

Resources:
 instance:
   Type: "AWS::EC2::Instance"
   Properties:
     ImageId: ami-82a458e0
     InstanceType: t2.large
     KeyName: !Ref key
     NetworkInterfaces:
       - DeviceIndex: 0
         SubnetId: !Ref subnet
         AssociatePublicIpAddress: true
```

```
      UserData:
        Fn::Base64: !Sub |
          <powershell>
          mkdir c:\inetpub\test
          curl https://s3-ap-southeast-2.amazonaws.com/test-aspx/default.aspx
-OutFile C:\inetpub\test\default.aspx
          add-windowsfeature web-webserver -includeallsubfeature -logpath
$env:temp\webserver_addrole.log
          new-website -name test -port 80 -physicalpath c:\inetpub\test
-ApplicationPool ".NET v4.5" -force
          remove-website -name "Default Web Site"
          start-website -name test
          </powershell>
      Tags:
        -
          Key: Name
          Value: !Ref serverName
```

- In the template you will see a "Description", "Parameters", and "Resources" sections.

- Parameters are the inputs you're prompted for when you launch the template.

- Resources are the service instances that will be created in the stack.

- Notice that the EC2 Instance has UserData in the template, the same user data that you pasted into the console earlier.

- Go ahead and create a stack from this template. If you need a hand follow these steps:

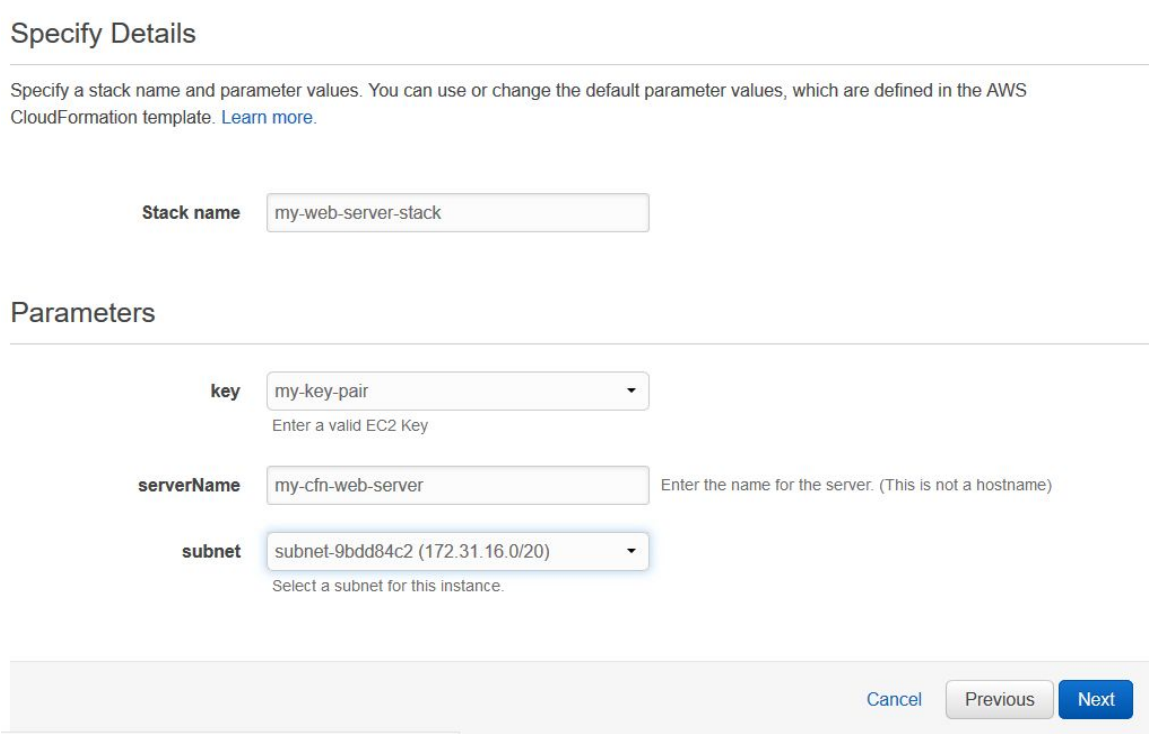
- Navigate to CloudFormation

- Select "Create Stack"

- Select "Upload a template to Amazon S3"

- Select "Browse..." and select the template file from your computer

- Select "Next"



- Enter "my-web-server-stack" for the "Stack name"

- Select your "Key Pair" for the "key"

- Enter a "serverName", let's call this one "my-cfn-web-server"

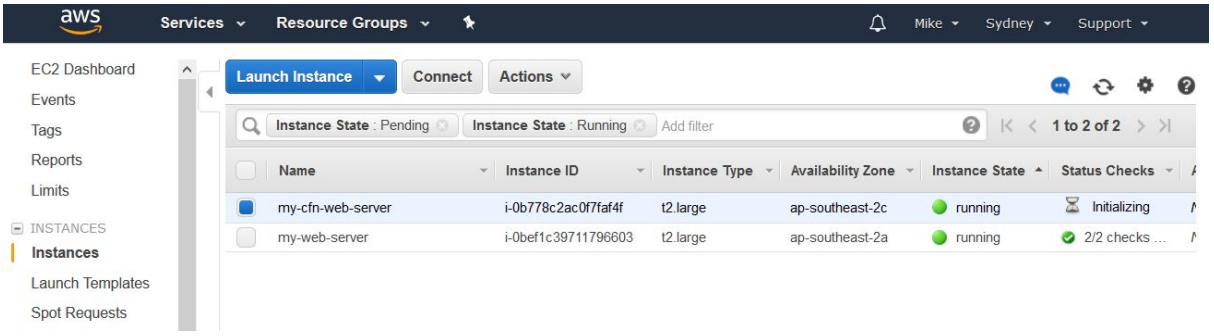
- For "subnet" choose any subnet in the "172." range.

- Select "Next"

- Scroll down and select "Next"

- Select "Create"

- Now lets navigate back to EC2 and see how our other build is going
- By now we should see 2 EC2 Instances building.



- It can take a few mins for the servers to build. This is mainly them enabling the Web Role and setting up the website.

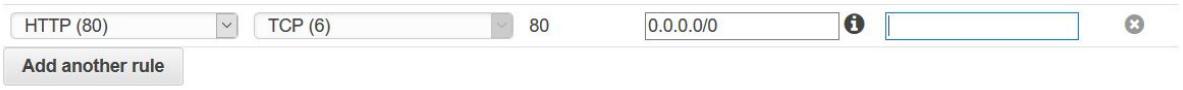
- While the CloudFormation template was excellent, it didn't enable port 80 for the Security Group. Let's fix this:

- Navigate to VPC

- Select 'Security Group' from the menu on the left hand side.

- Locate and select the security group with the description 'default VPC security group'.

- Select the 'Inbound Rules' tab, and select 'Edit'



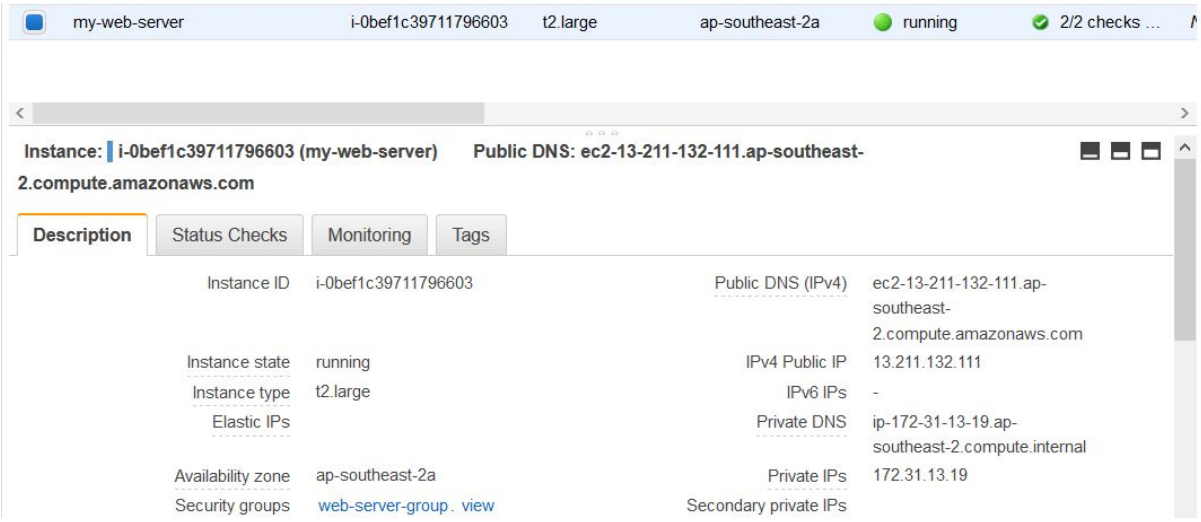
- In the new entry, select 'HTTP' for 'Type', and add '0.0.0.0/0' to the 'Source'

- Select 'Save'

- Ok, let's take a look at our handy work!

- Select the first server we created "my-web-server"

- In the details towards the bottom of the page find and copy the "Public DNS" value for the server.



- Enter this value into a new tab of your web browser.
- Do you see a webpage?

- No need to RDP into the server, the job is done.

- In a few minutes, do the same with your Cloudformation server.

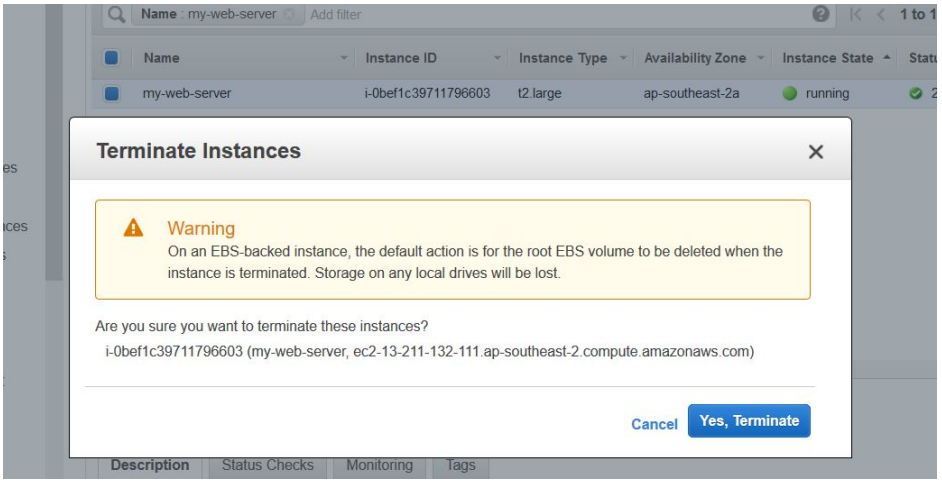
- You should see a webpage there too.

- Discuss: How could we get this build to be even faster?

Cleanup the Resources

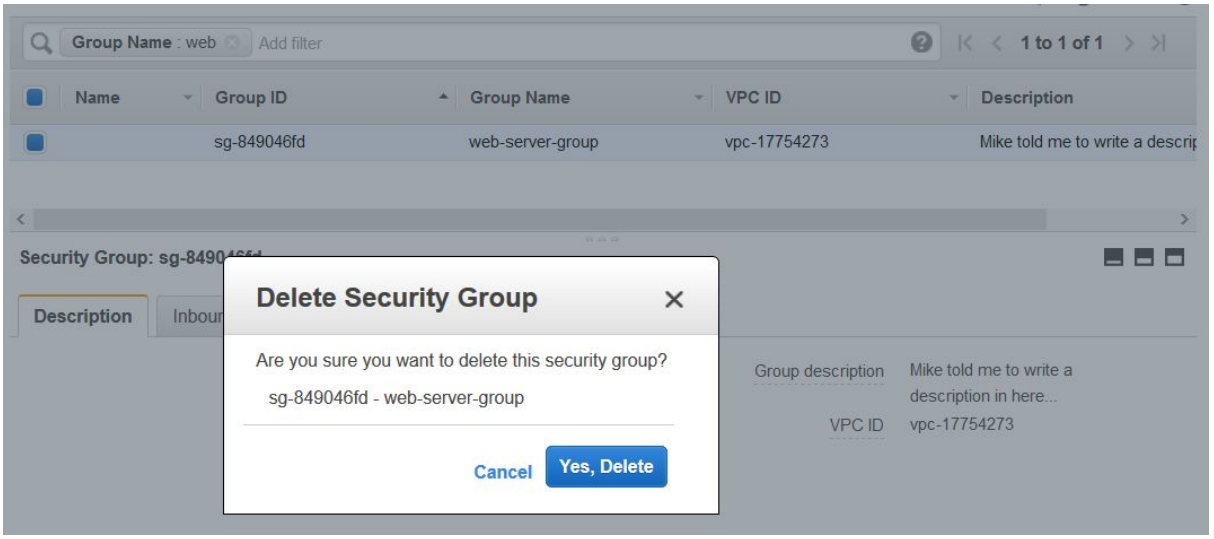
- We don't need these servers anymore. Let's clean up:

- Navigate to EC2
- Select the instance called "my-web-server"
- Select "Actions", "Instance State", "Terminate" and "Yes, Terminate"



- From the menu on the left hand side, select "Security Groups"

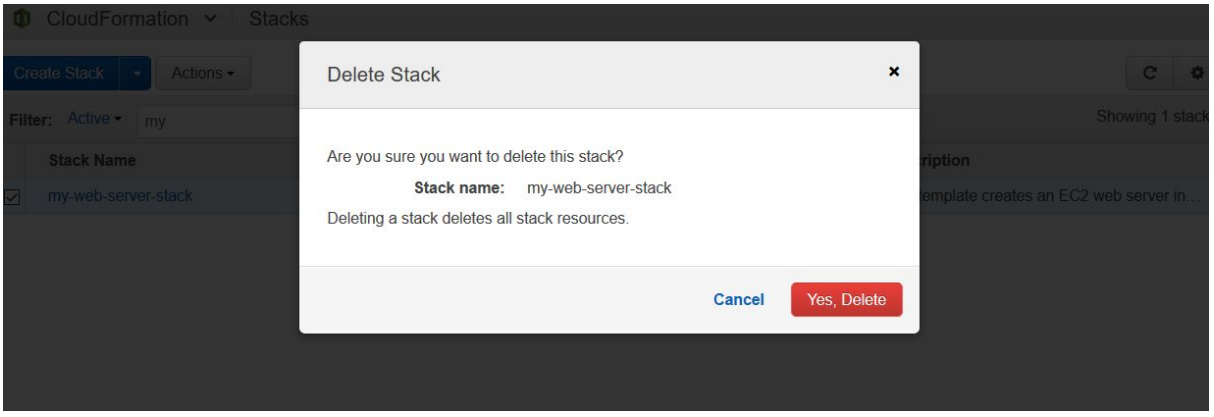
- Look for a Security Group with the GROUP NAME (note this is not the "Name" column) of "web-server-group", select "Actions", "Delete Security Group", and "Yes, Delete".



- Cleaning up the CloudFormation server/stack is much easier

- Navigate to CloudFormation

- Select your web server stack, then "Actions", "Delete Stack", "Yes, Delete"



Extra Challenge

While we wait for everyone to complete this session….

- What other things could you do during a boot-strap?
- Could you pass in credentials to do tasks such as domain join?

END

Session 5 - AWS Systems Manager

In this session we will get introduced to AWS Systems Manager (SSM).

Take the following steps, one at a time. There is plenty of time to complete the session and ask any questions along the way.

Steps

- First we're going to set up a (small) fleet of servers to manage.

- Navigate to CloudFormation

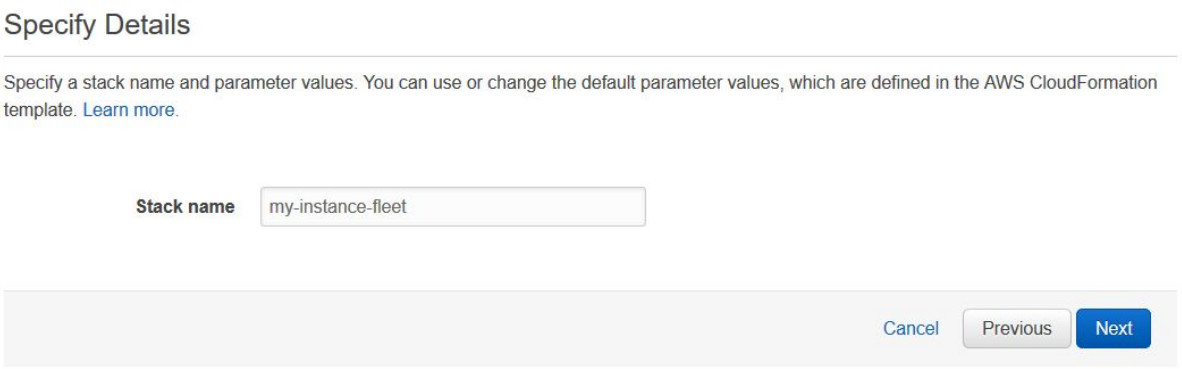
- Select "Create Stack"

- Select "Upload a template to Amazon S3" and "Browse..."

- With the resources for this session find "three-server-instances.yaml"

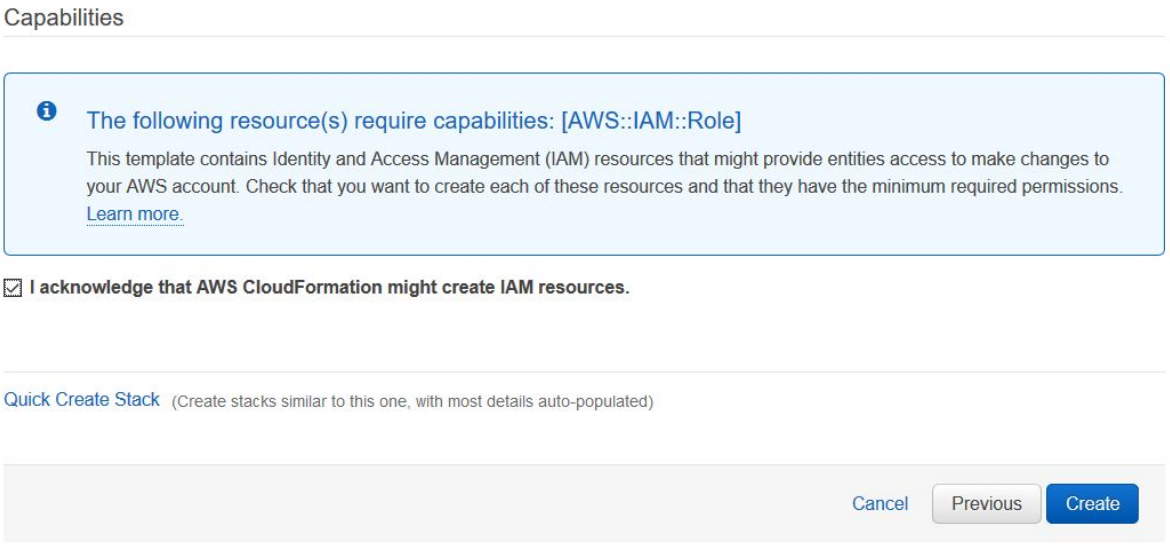
- Select "Next"

- For the stack name enter "my-instance-fleet"



- Select "Next"

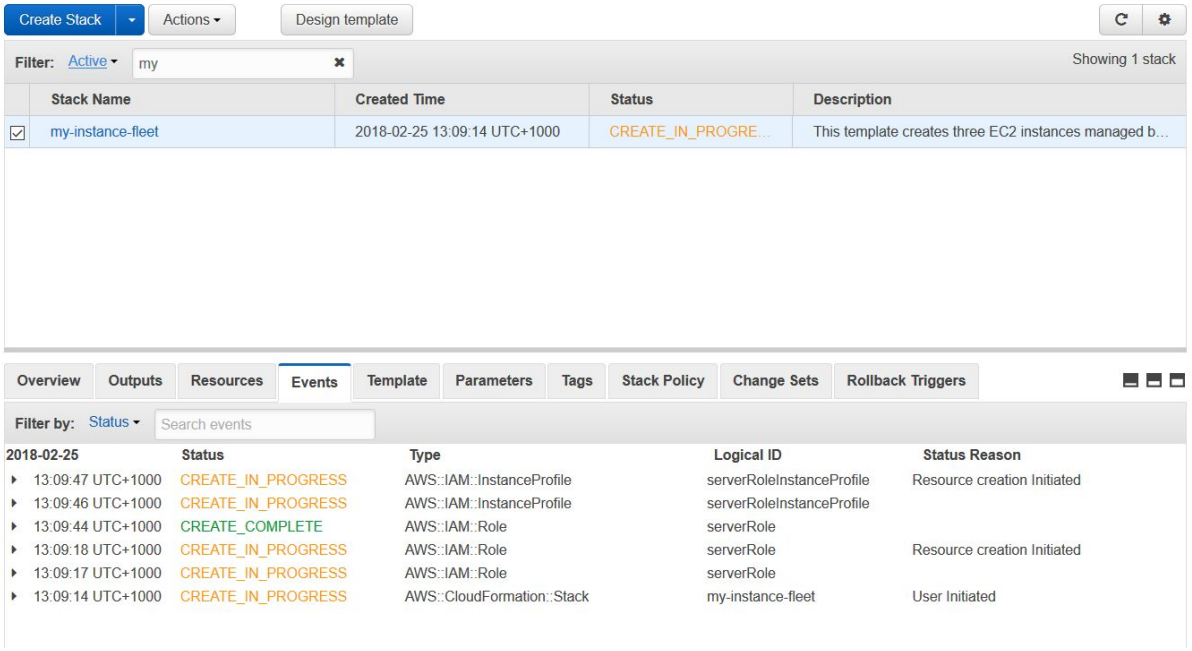
- Scroll to the bottom and select "Next"



- As this template will adjust the security of your account you will now need to check the box for "I acknowledge that AWS CloudFormation might create IAM resources."

- Select "Create"

- Wait a minute or two while the instances build. You can watch them build from the "Events" tab in the CloudFormation stack page, and from the Instance list in the EC2 page.



- The stack can take longer than you might expect to create the "AWS::IAM::InstanceProfile", but things should speed along after that step is complete.

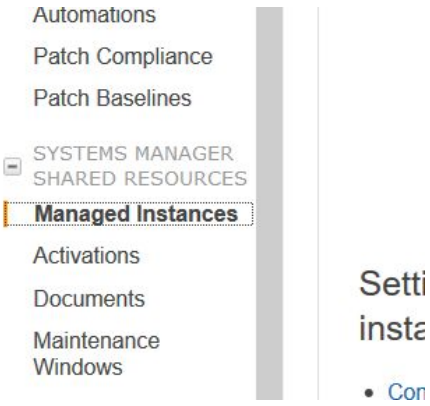
- Still waiting? Have a look at the template you are building. In some ways its more complex than other templates we've built.

...

- Now the instances are built, let's check to make sure that SSM can 'see' them.

- Navigate to EC2

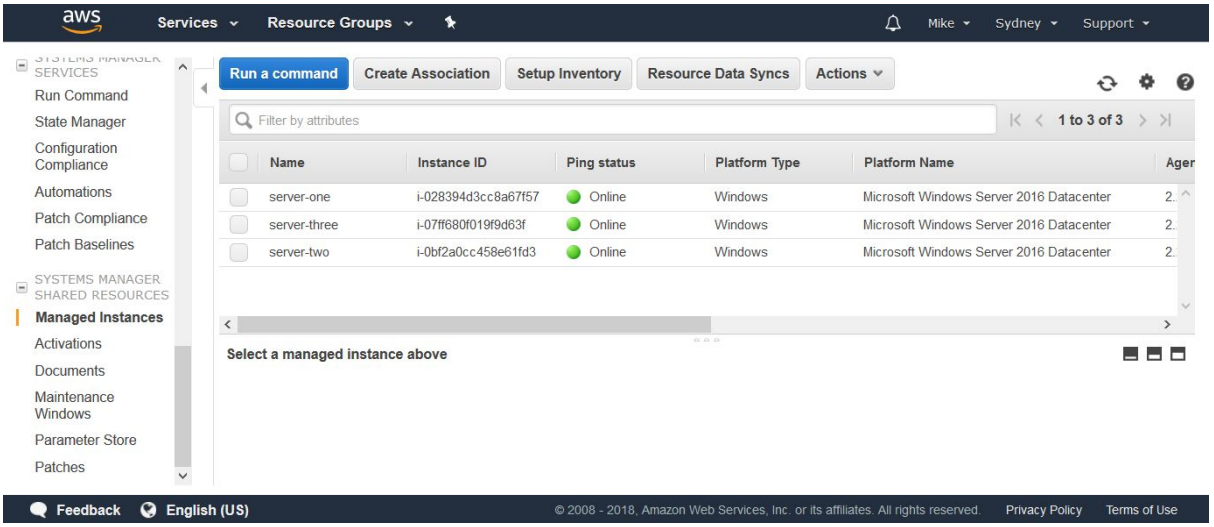
- From the menu on the left hand side, scroll down and select "Managed Instances" under the heading "SYSTEMS MANAGER SHARED RESOURCES".



- If there is nothing listed yet, the SSM service might not have discovered the Instances yet. Wait a moment.

- When we created the Instances (via CloudFormation) we gave them the security rights to communicate with SSM. We are now waiting for the SSM agent on the server to poll the services.

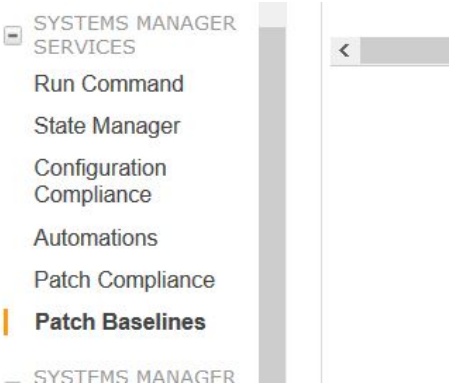
- Refresh the page (maybe a few times) until you see all three servers listed.



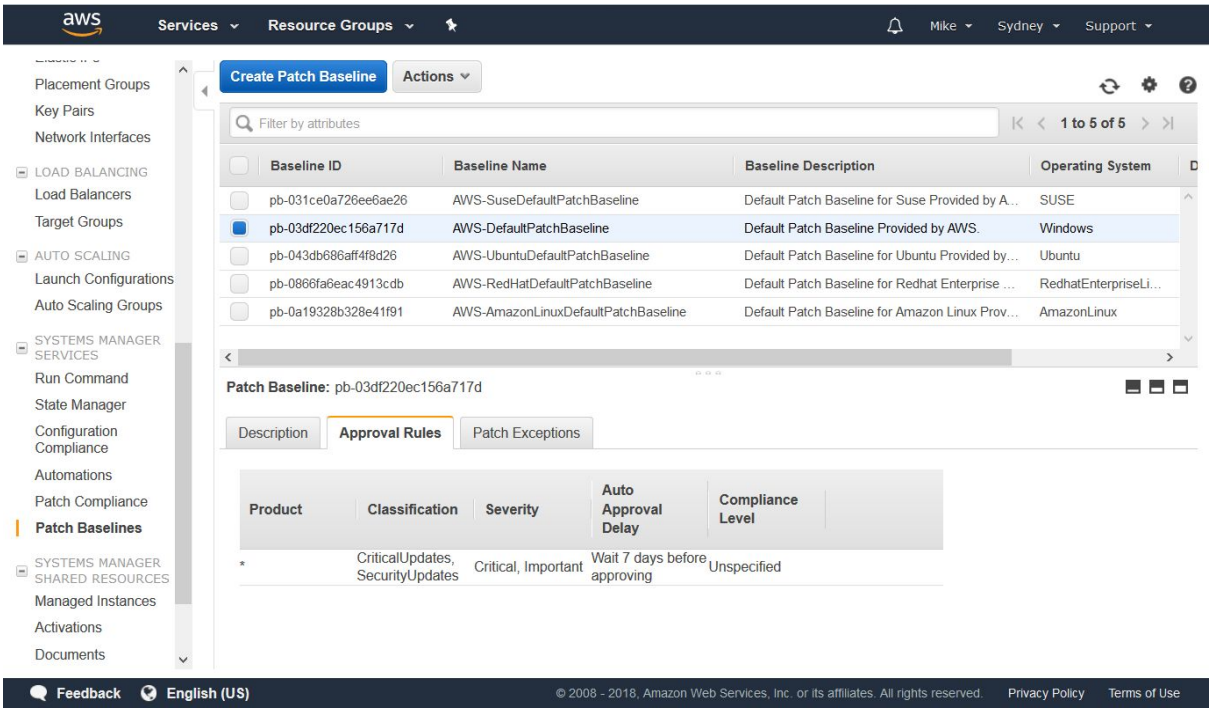
SSM - Check server patch status

- First let's check the patch status of our Instances against the default Patch Baseline.

- To see the default baseline select "Patch Baselines" from the menu on the left hand side.

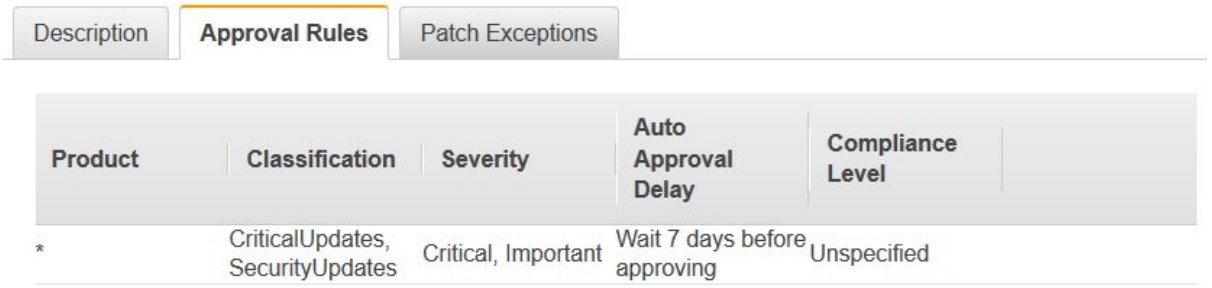


- You will see several Default Baselines provided by AWS. Select the one with the name "AWS-DefaultPatchBaseline" as this applies to Windows.



- In the lower section of the screen select "Approval Rules"

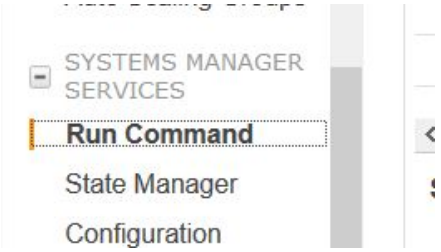
- You can see that this Baseline defines all patches with:
  - Classification of "CriticalUpdates, SecurityUpdates"
  - Severity of Critical, Important
  - Older than 7 days



- Let's check our servers against this baseline.

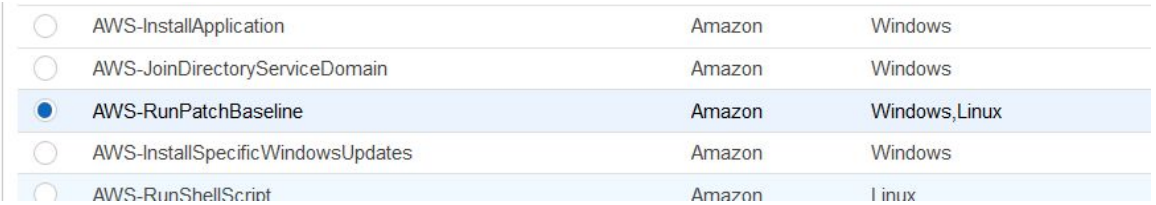
- Normally we would do this using a "Maintenance Window", or “State Manager” that checks periodically, but today we will run a one off check using "Run Command"

- Select "Run Command" from the menu on the left hand side.

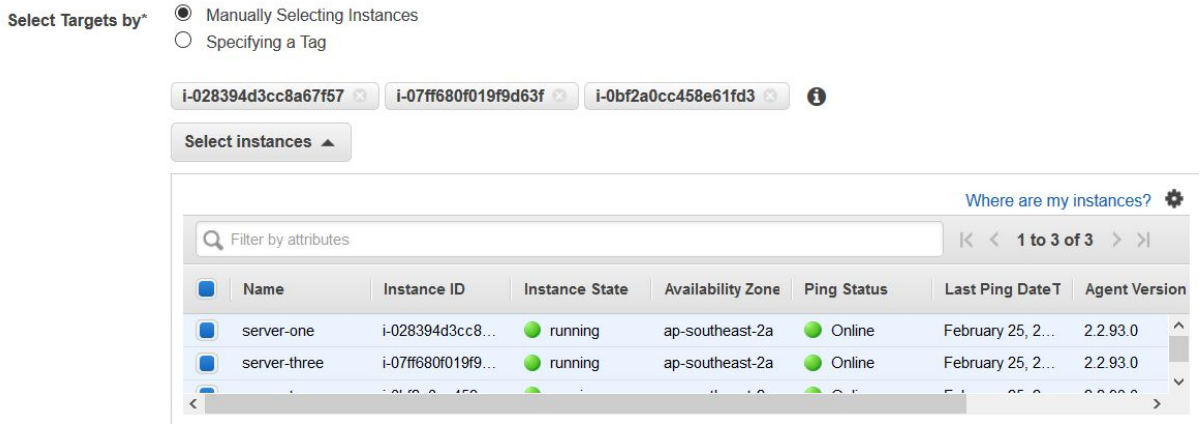


- Select "Run a command"

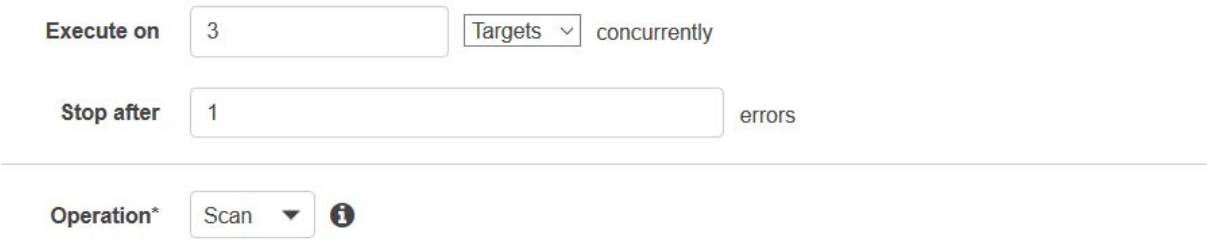
- Under "Command document" select "AWS-RunPatchBaseline". This is a prewritten document to check any type of instance against a Patch Baseline.



- Scroll down and select "Manually Selecting Instances"



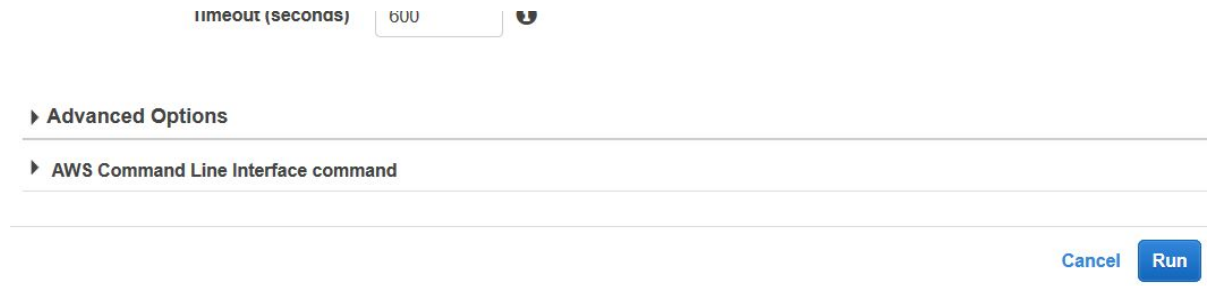
- Select "Select instances" and select all of your three instances.



- Enter "3" for "Execute on", this specifies how many instances of the document should run at once across our fleet.

- Enter "1" for "Stop after" ... "errors".

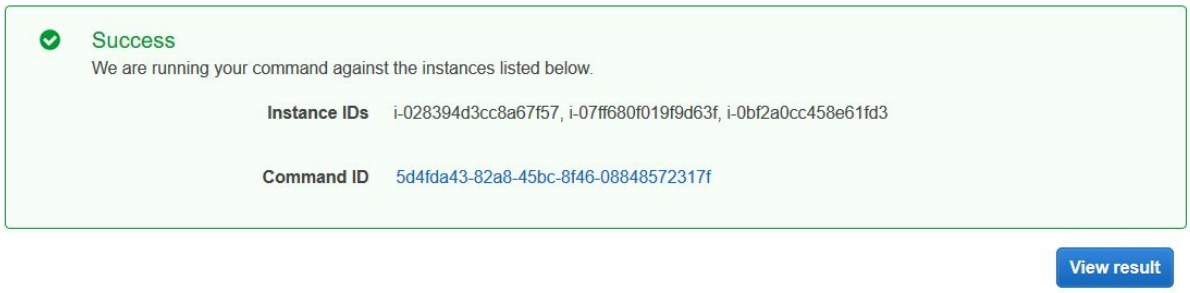
- For "Operation" select "Scan". We are not going to install missing patches, but rather scan for missing patches.



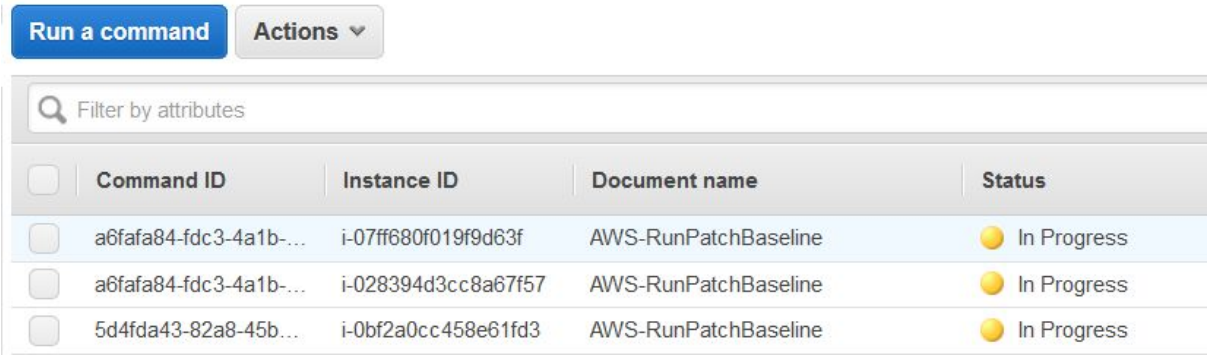
- Scroll to the bottom and select "Run"

- Ensure on the next page that there are entries under "Instance IDs"

- Select "View result"

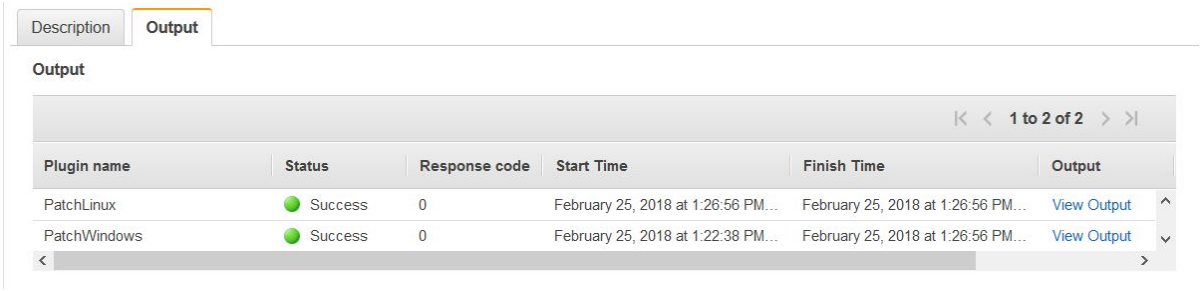


- You should see a list of commands "In Progress", the list has a refresh button on it so you can watch the progress.



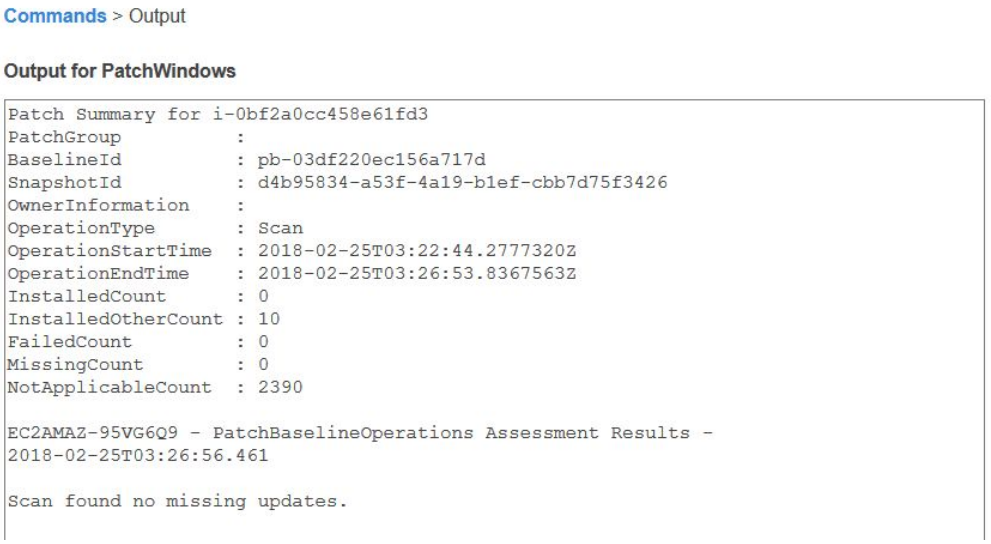
- When any of the commands are complete, shown as "Success" you can view the output.

- Select the command from the list, in the lower section of the screen select the "Output" tab.



- In the output list you should see two entries. The command/document that we ran operates on both Windows and Linux OS's. There should be one output per OS type, and only any useful information in the Windowseat man output.

- You will (probably) see that "Scan found no missing updates."

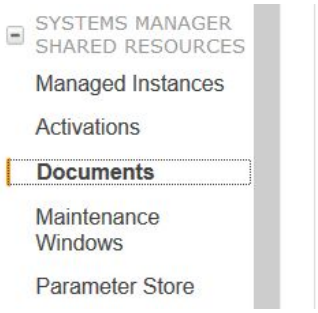


- Great.

SSM - Run our own command/document

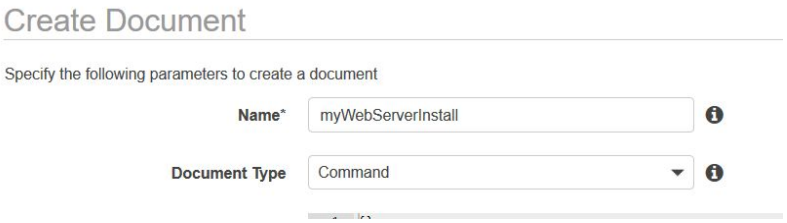
- Now let's create our own document, and run a bigger command.

- Navigate to EC2, scroll down the left hand menu and select "Documents"



- Select "Create Document"

- Enter "myWebServerInstall" into the "Name"
- Make sure that "Document Type" is set to "Command"



- In the resources for this session you will find a file called "ssm-document.json". Open this in a text editor and copy all of the content to the clipboard.

- Under "Content" delete the curly braces "{}" and paste in the contents of your clipboard.

- There will be some time in a moment to review the document.

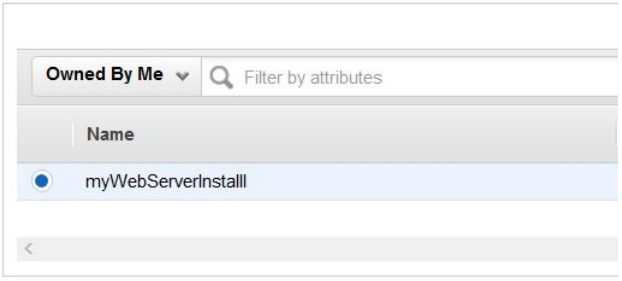
- Scroll down and select "Create Document", then "Close"

- Let's run this document as a command against our fleet.

- Select "Run Command" from the menu on the left hand side.

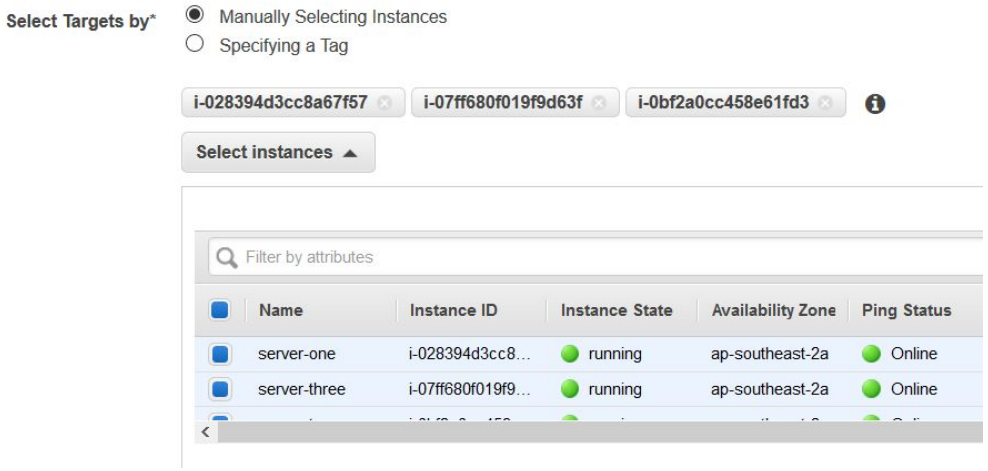
- Select "Run a command"

- Under "Command document" change the dropdown filter to "Owned By Me"



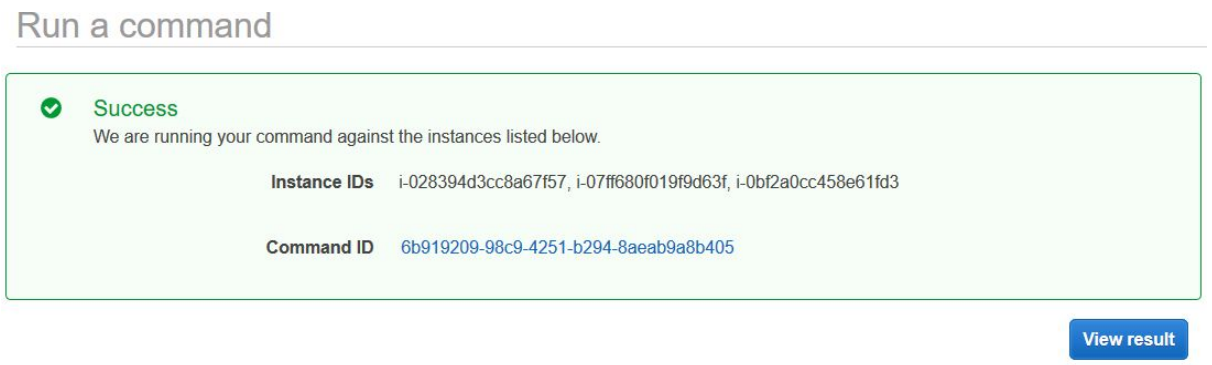
- Select your document "myWebServerInstall"

- Again select "Manually Selecting Instances" and select all your instances from "Select instances"



- Again select "3" and "1" for "Execute on" and "Stop after"

- Scroll down and select "Run", ensure the instances are listed, and select "View result"

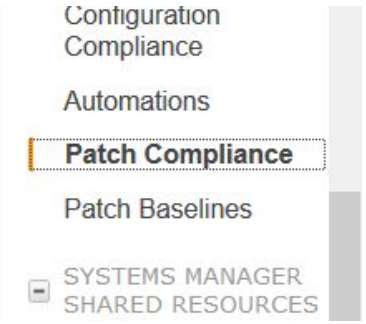


- This might take a couple of minutes.

….

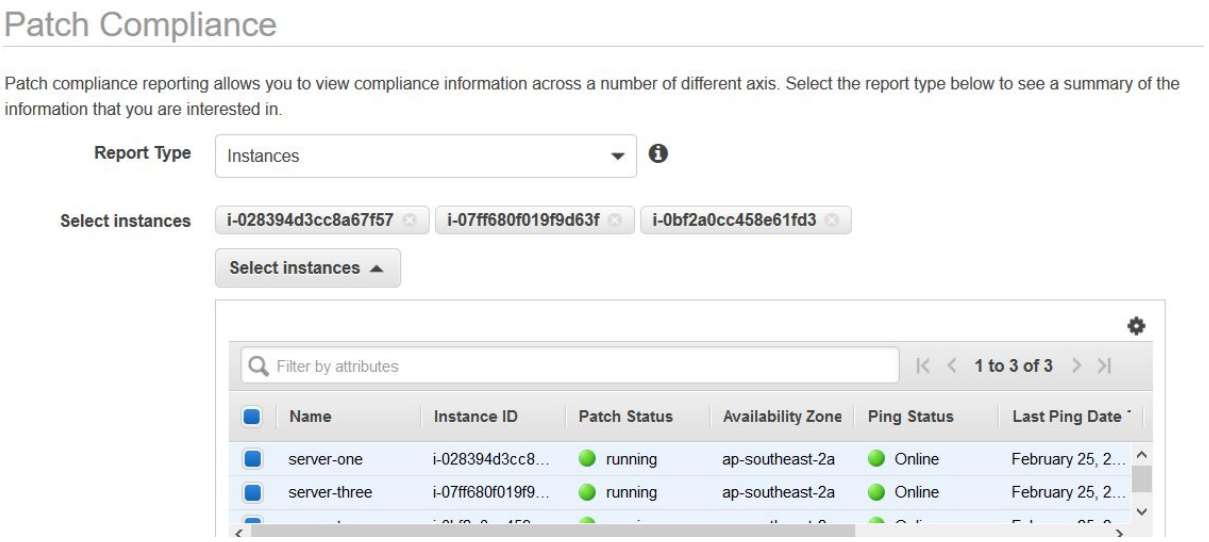
- Let's go and take another look at the result of the patch baseline we did.

- From the menu on the left hand side select "Patch Compliance"

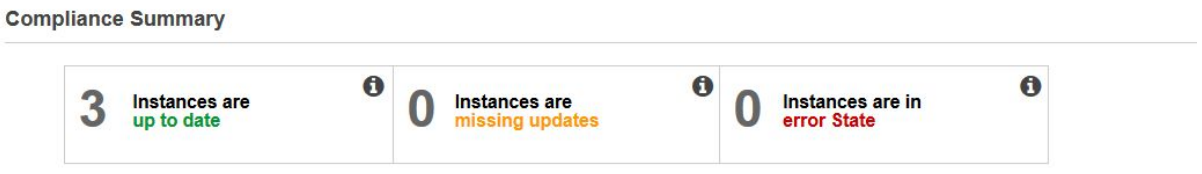


- This page will show a report on the patch compliance of our managed fleet.

- In the "Select instances" dropdown, select all our instances



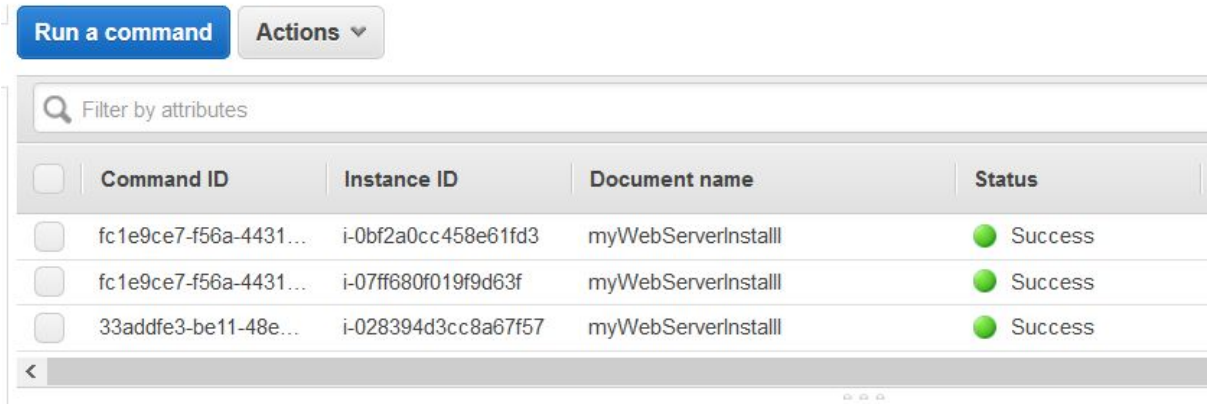
- The dashboard should update with a review of the current patch status.



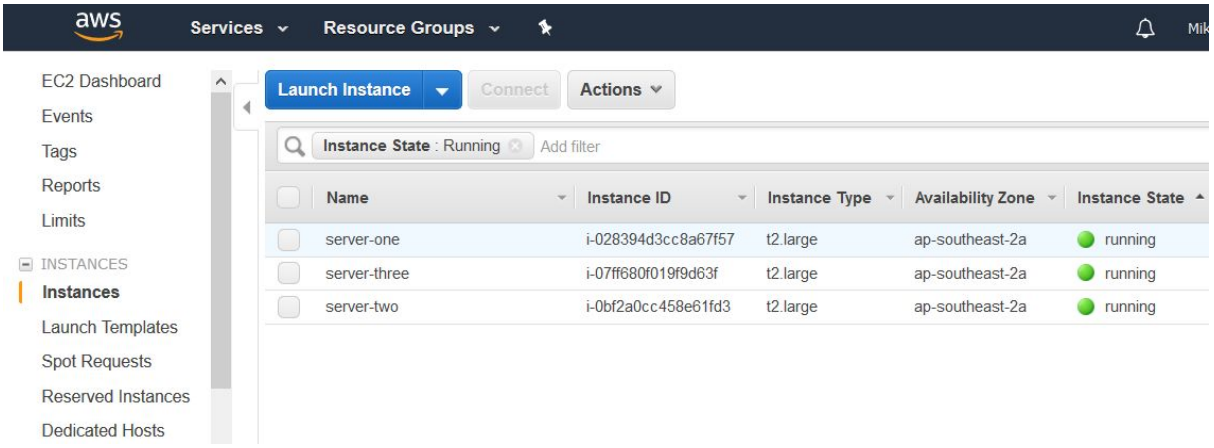
- This is a useful tool, especially if you are running "AWS-RunPatchBaseline" on a regular basis.

- To catch up on the status of our web server installs, select "Run Command" again from the left hand menu.

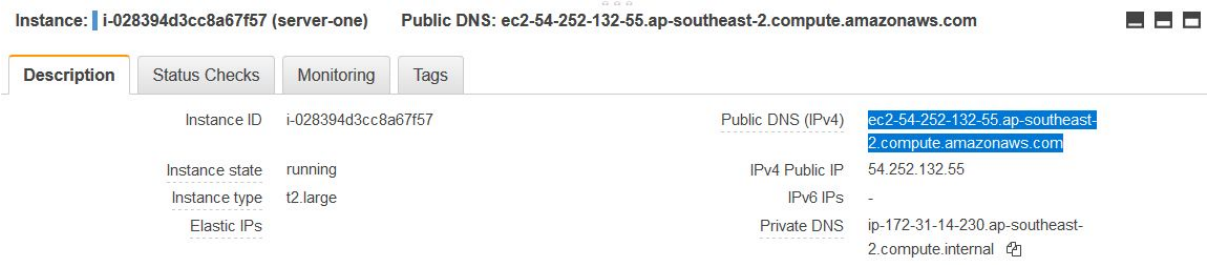
- When the commands are showing as "Success"



- Navigate to EC2, Instances



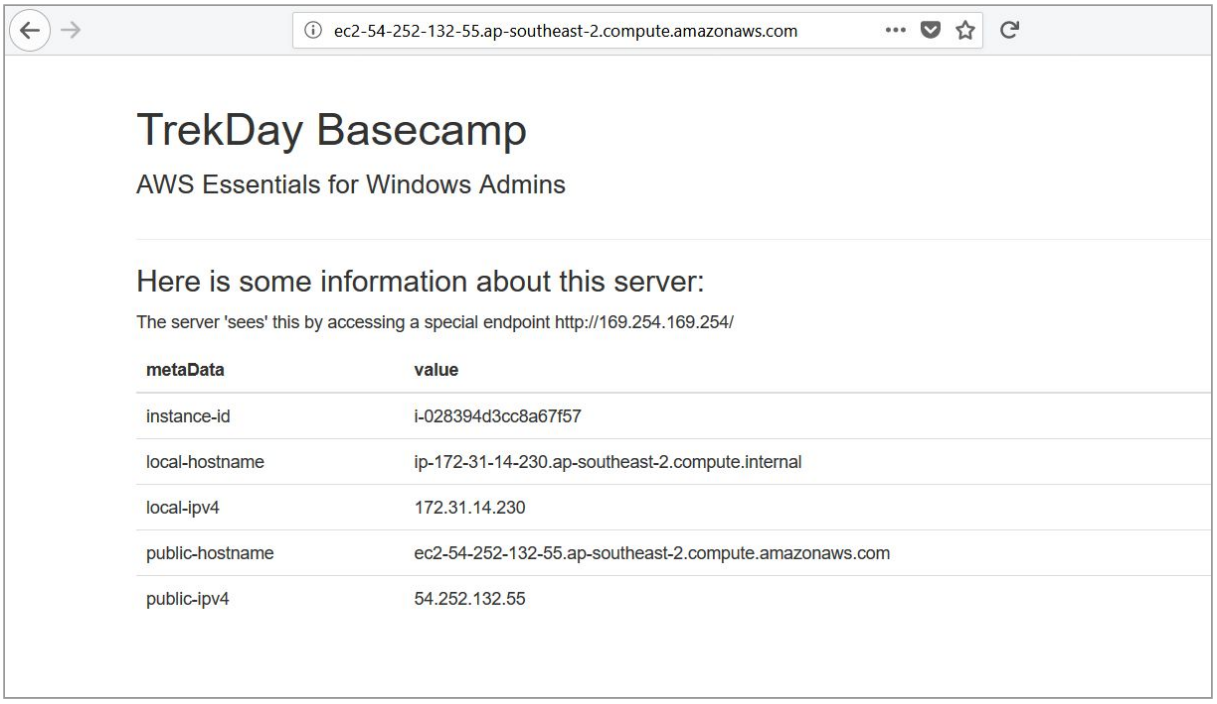
- Locate the "Public DNS" for the instances and test the website from the details section at the bottom of the screen.



- You may have to wait a few moments after the command is complete before the website appears.

- Issue: Sometimes the run commands don't show in the list. Try creating them again.

- Open a new tab in your browser and navigate to the Public DNS name of one of the instances:



- It's our website again.

- Fancy logging into the Instance via RDP? Well you can't we didn't use a Key Pair, this server is 100% not a pet.

- No need to clean up, leave the server fleet there, we'll be using it in the next session.

Extra Challenge

While we wait for everyone to complete this session….

- Let's talk about how we can use SSM to get logging data from the EC2 Instance into CloudWatch Logs

END

Session 6 - Security

In this session we will get introduced to Amazon Inspector.

Take the following steps, one at a time. There is plenty of time to complete the session and ask any questions along the way.

Steps

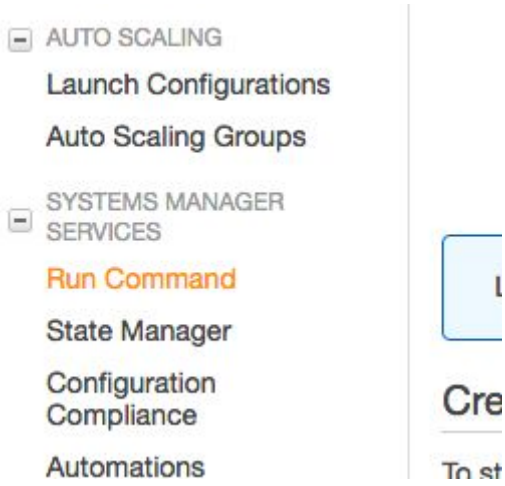
- Do you still have your small fleet of servers?
- If you deleted the stack after the SSM session, now would be a good time to recreate it!

- We're going to start this session off, back in SSM.

Run an SSM Command

- Navigate to EC2

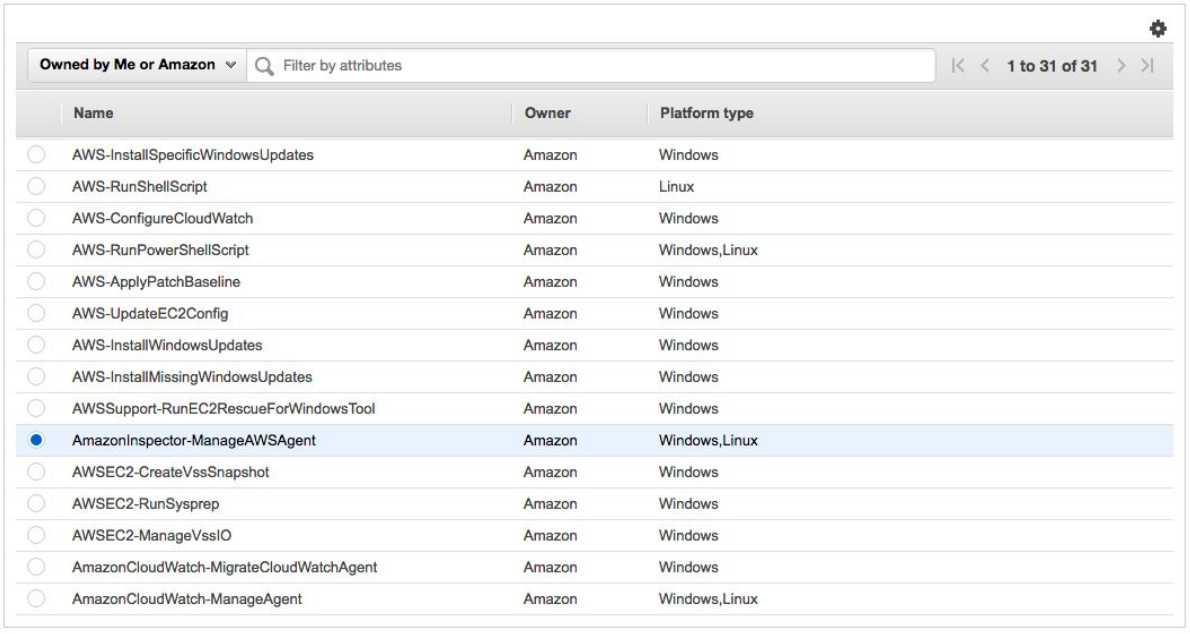
- From the menu on the left hand side select "Run Command"



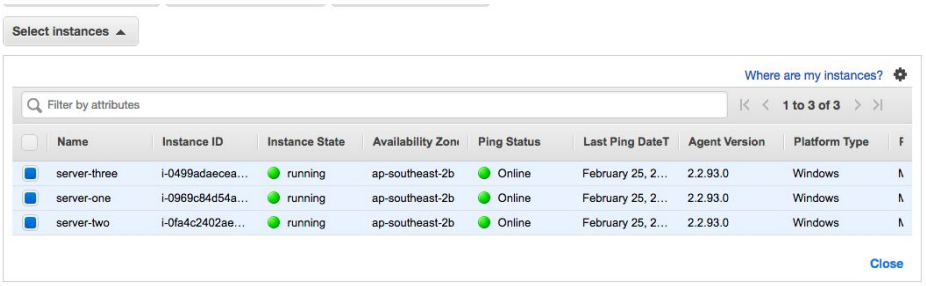
- We're going to install the "AWS Inspector" agent on our fleet.

- Select "Run a command"

- Select "AmazonInspector-ManageAWSAgent"

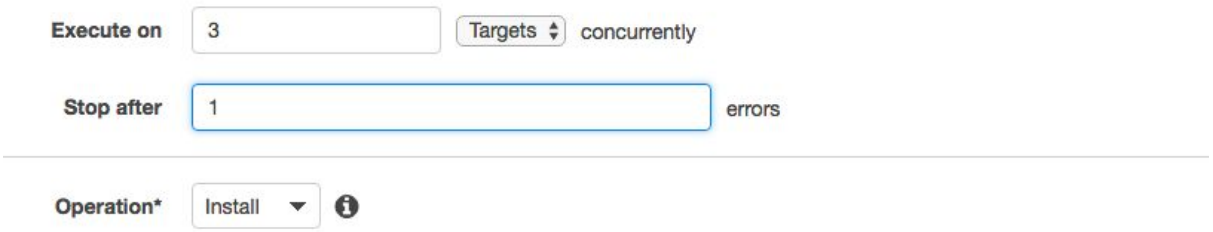


- From the "Select instances" dropdown, select all your instances

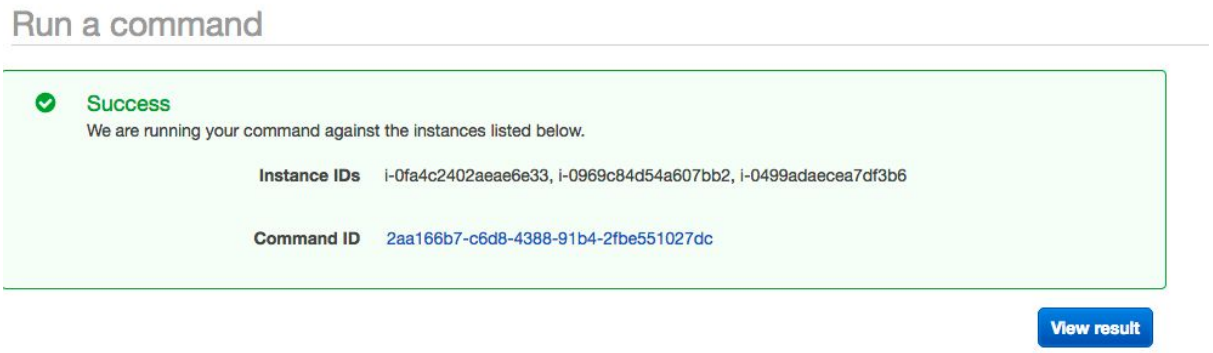


- Enter "3" and "1" for "Execute on" and "Stop after"

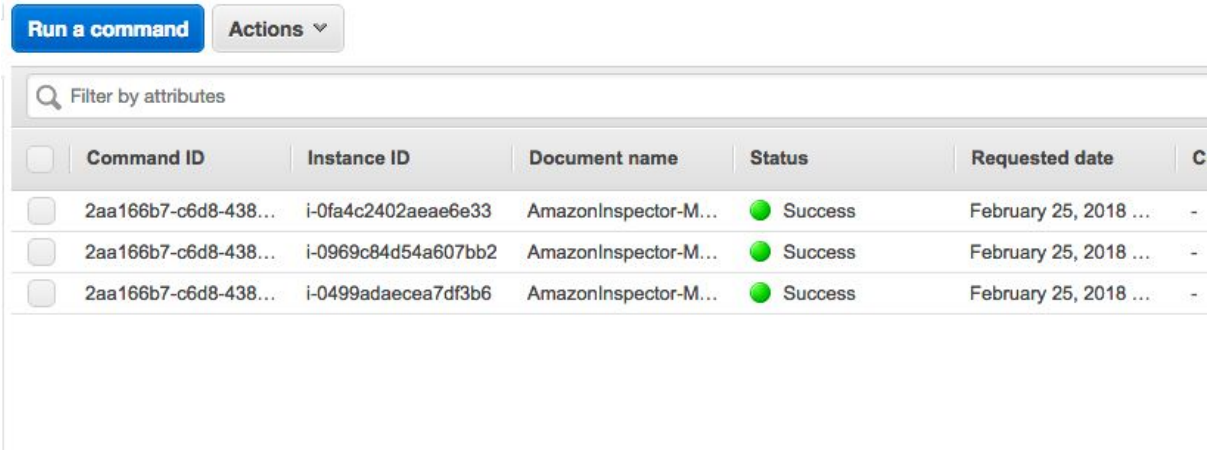
- For "Operation" ensure that "Install" is selected



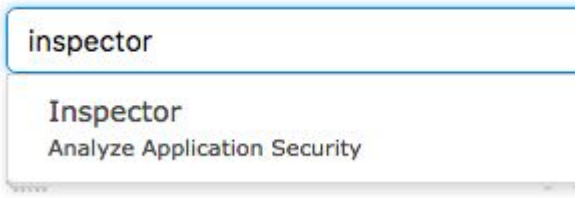
- Scroll down and select "Run", then "View result"



- When the command is a "Success" the Amazon Inspector Agent is installed on the instance.



- Navigate to "Amazon Inspector"



- Select "Get Started"

- You should now see the "Get started with Amazon Inspector" page

- Notice that an IAM role will be created for us called "AWSServiceRoleForAmazonInspector", AWS creates this on our behalf.

- Select "Next"

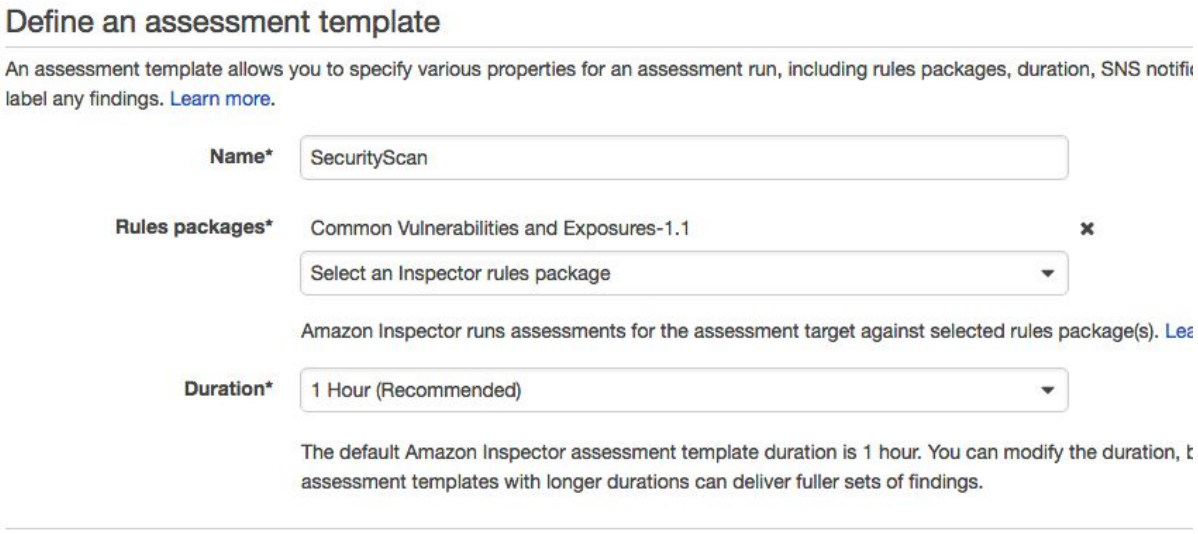
- Under "Define an assessment target"

- Enter a name for the assessment "WindowsDevAssessment"

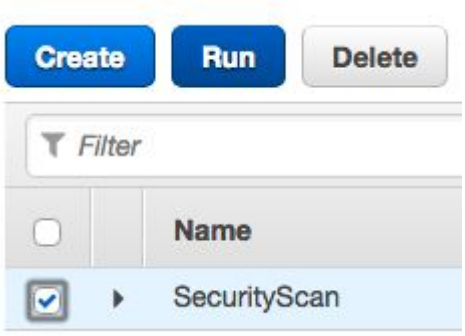
- Under Tags, select Key "Environment" and Value "dev"
- Select "Next"

- Under "Define an assessment template"
- Enter a name of "SecurityScan"

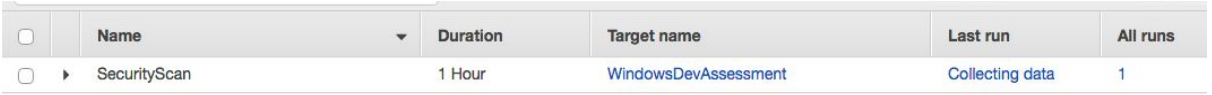
- Expand the "Rules packages" dropdown and review the types of packages that can be run.
- For now select "Common Vulnerabilities and Exposures-1.1"
- For "Duration" select "1 Hour (Recommended)"
- Select "Next" and "Create"



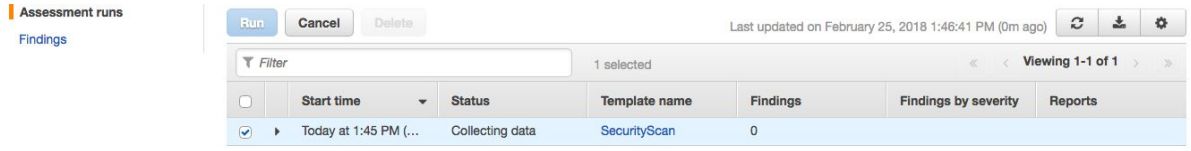
- Now that we have defined an assessment, let's run it
- Select "SecurityScan" from the list
- Select "Run"



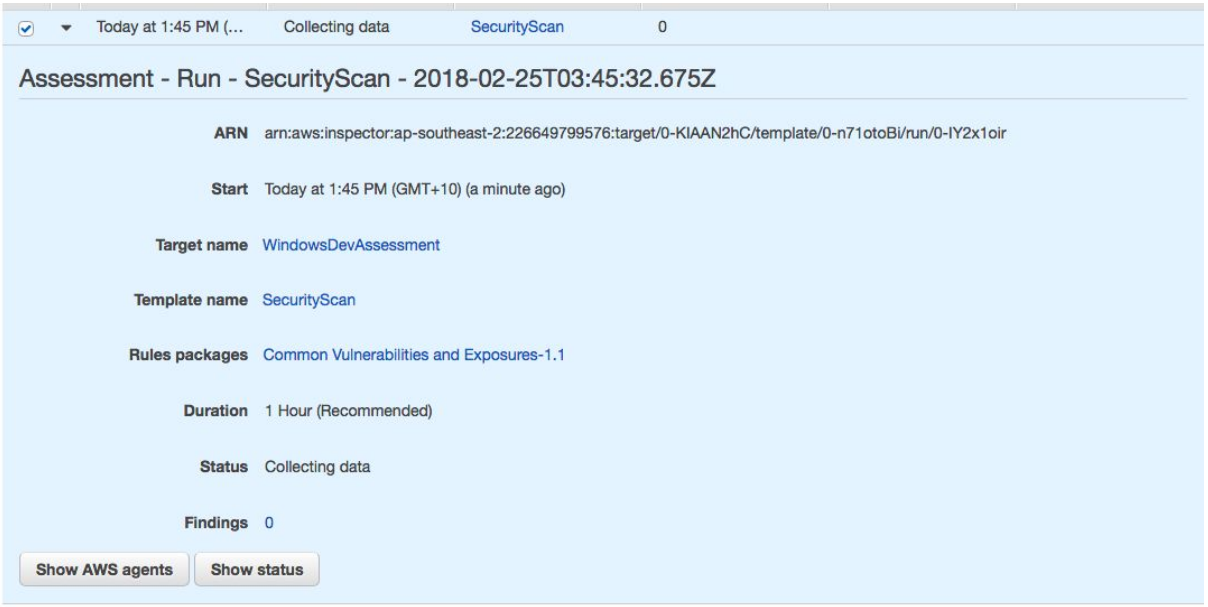
- Notice that the "Last run" column shows "Collecting data"



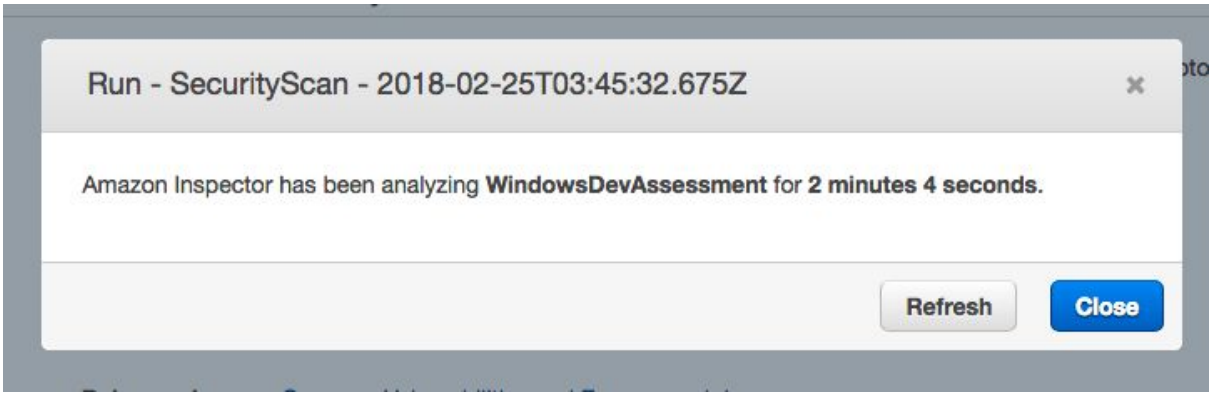
- If you navigate around while the run is collecting data, and want to check on the progress navigate to "Assessment runs" on the left hand side and expand the row entry to see the "Status"



- Click "Show status" for a detailed report on the scan progress.



- The scan will take some time, so lets come back to this later today.



END

Session 7 - Serverless

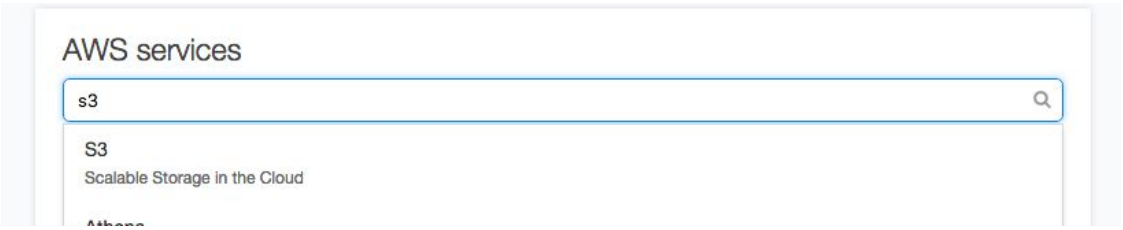
In this session we will get introduced to the concept of “Serverless” event driven architectures.

Take the following steps, one at a time. There is plenty of time to complete the session and ask any questions along the way.

Creating S3 Buckets

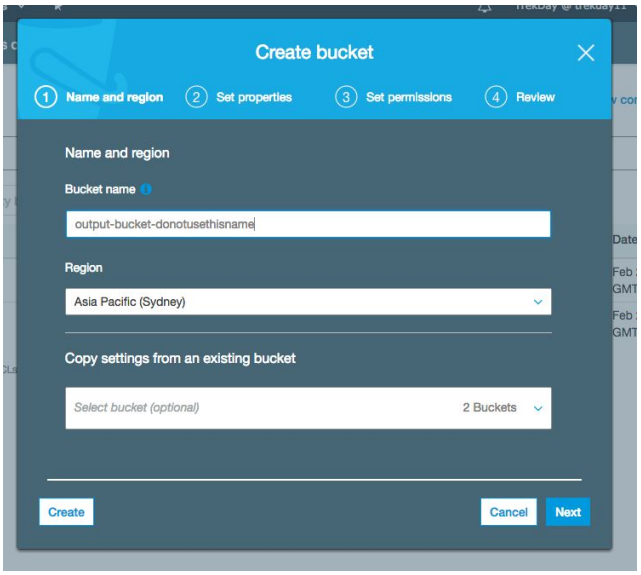
- For this workflow we will need a couple of S3 buckets

- Open the AWS console, make sure you are in Sydney, and navigate to S3



- Select Create bucket

- Enter a bucket name (this must be globally unique) don't use the same name as in the screenshot, and don't use the same as your neighbour!



- Tip: Enter the name "input-bucket-" then add some pseudo random characters by mashing the keyboard.

- The S3 console page is global, so you need to select a region to create the bucket. **Make sure this is Sydney.**

- Select "Create"

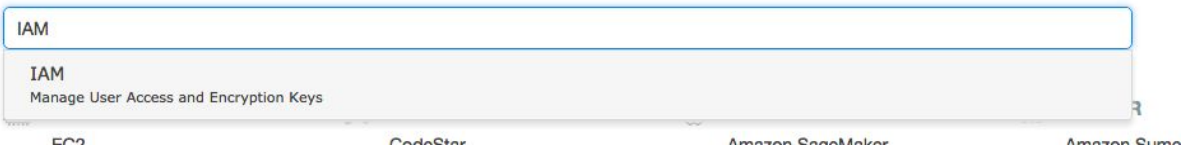
- Repeat this step and create a "output-bucket-xxxxxxxx"

- Make a note of the bucket names.

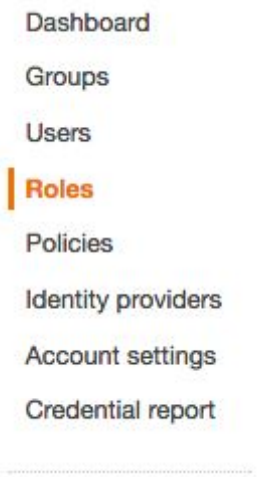
Creating an IAM Role

- Before we create our Lambda function, we need to create a security profile (called an IAM Role) that will give the function all the necessary permissions it needs to run.

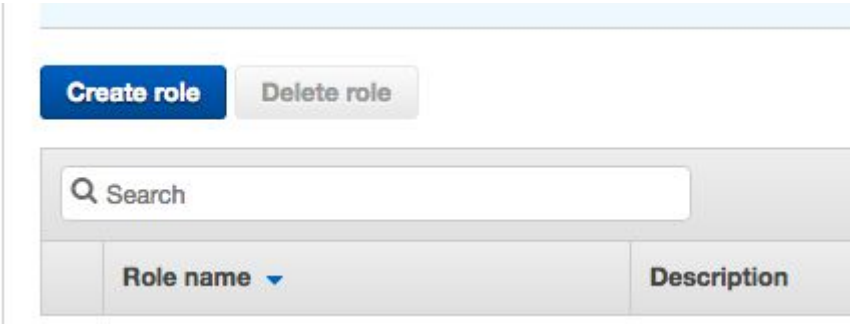
- Navigate to IAM (Identity and Access Management)



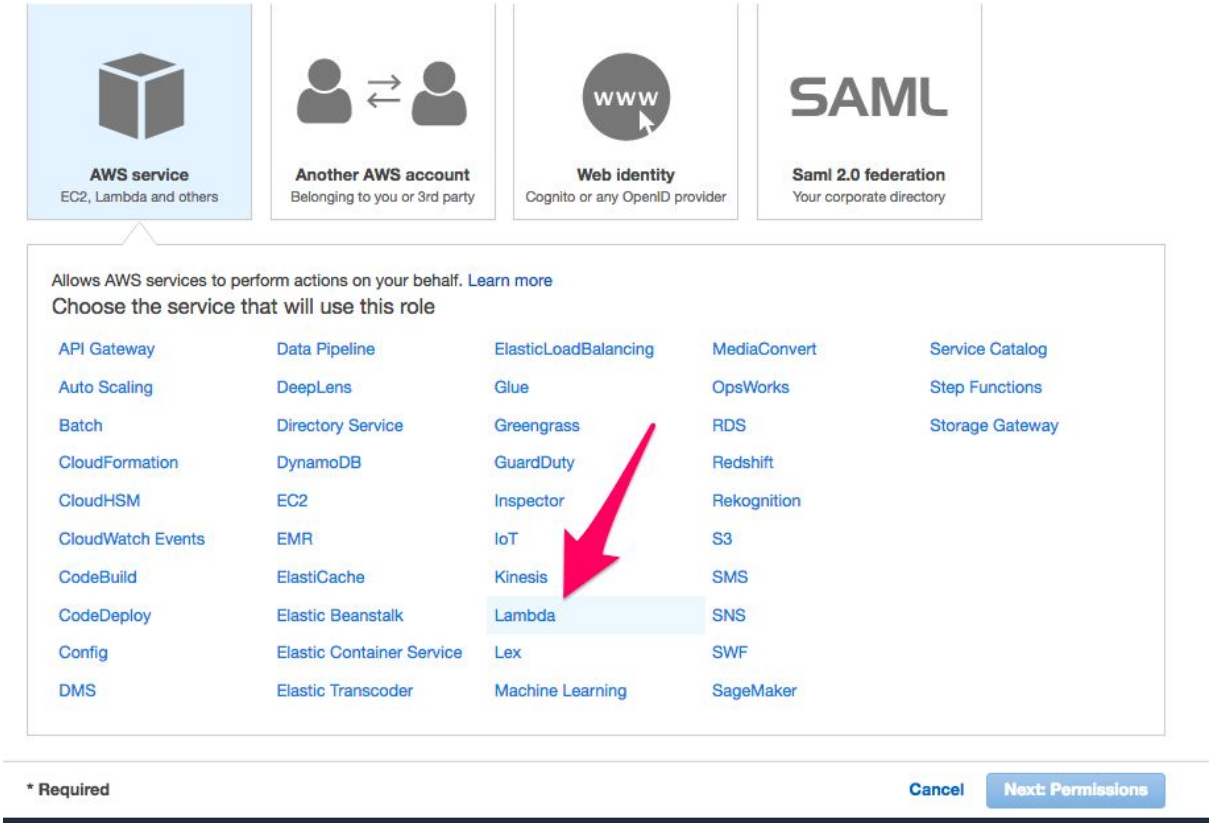
- Select "Roles"



- Select "Create role"

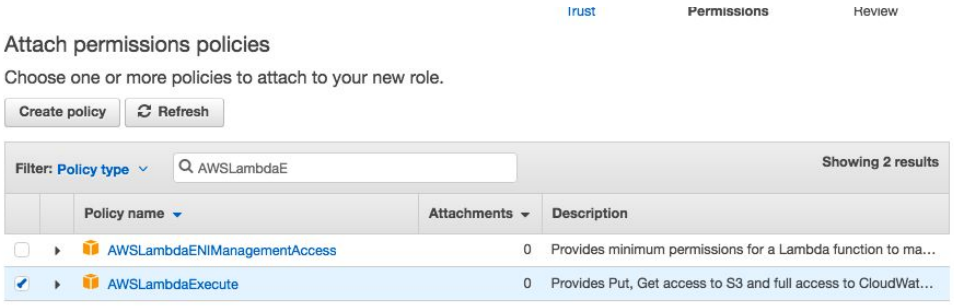


- Ensure that "AWS Service" is selected, then select "Lambda", and "Next: Permissions"



- In the policy type search box, type "Lambda", and scroll to "AWSLambdaExecute"

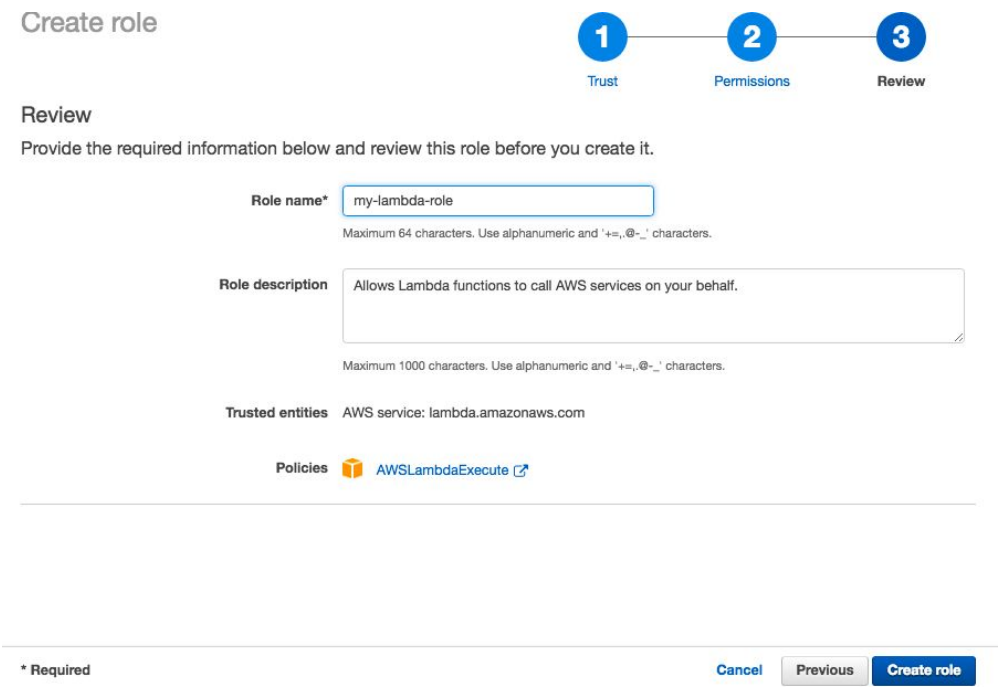
- SELECT THE CHECKBOX next to the policy name (if you select the policy name itself, you will be sent elsewhere!)



- Select "Next Review"

- Enter a name for your role "my-lambda-role"

- Select "Create role"

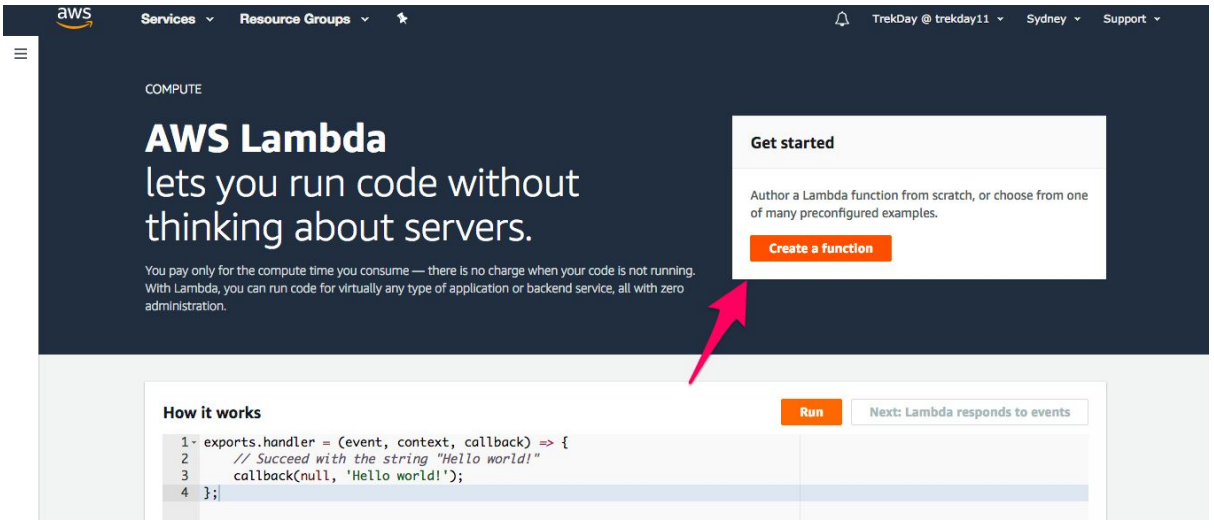


Create a Lambda Function

- Now lets create our Lambda function

- Navigate to Lambda

- Select "Create function"



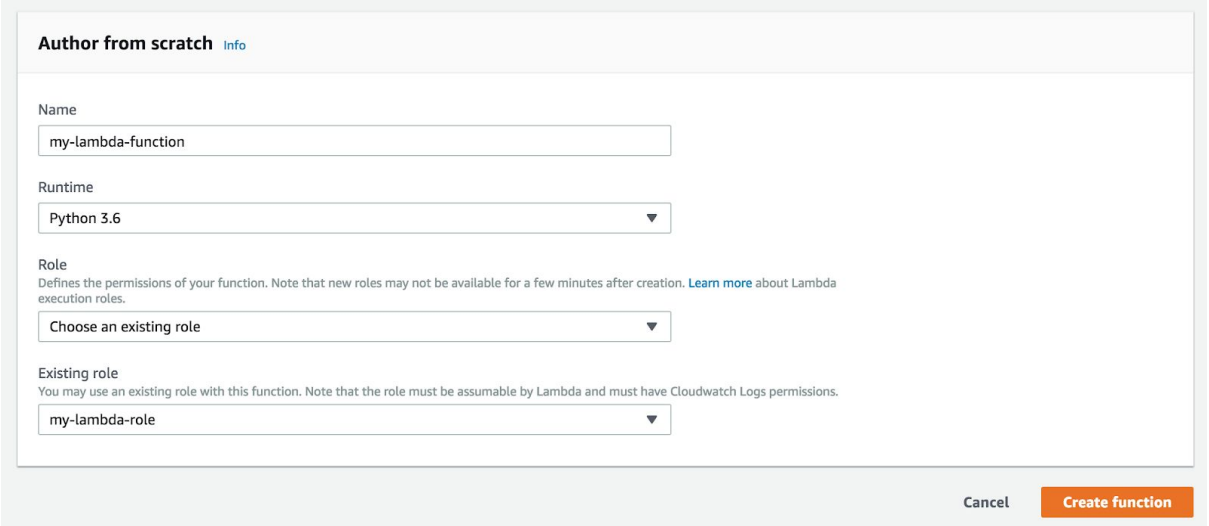
- Make sure that "Author from scratch" is selected

- Enter a function name: "my-lambda-function"

- In the Runtime selection, select "Python 3.6" ***(Notice the other runtimes you could use when writing your own functions.)***

- In Role, select "Choose an existing role" and then select in the "Existing role" selection to find "my-lambda-role"

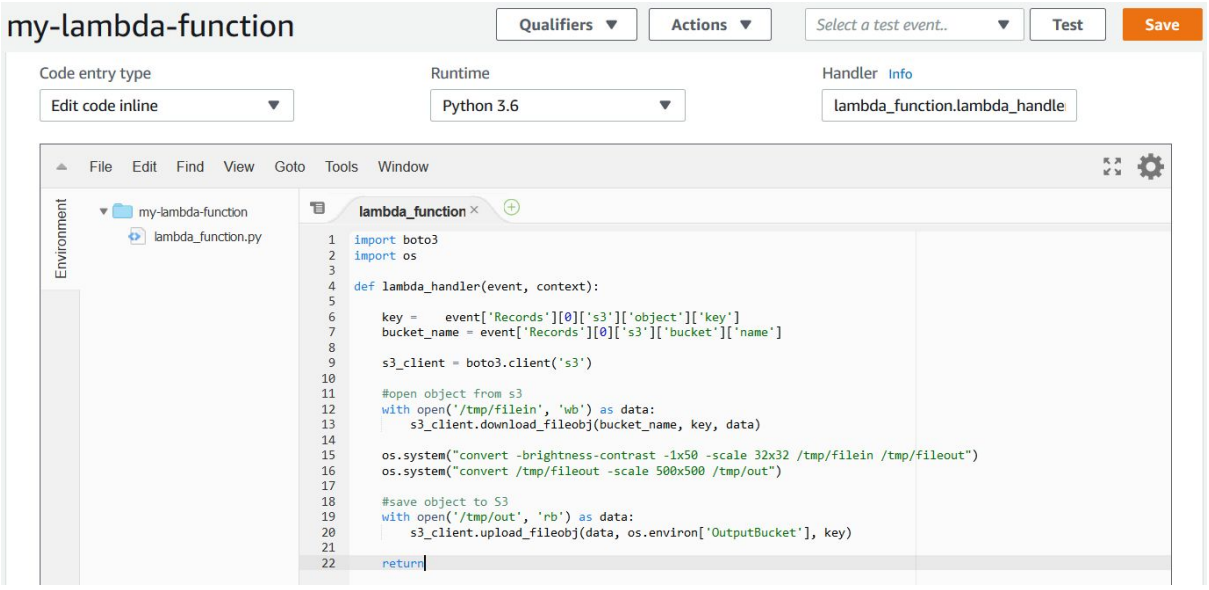
- Select "Create function"



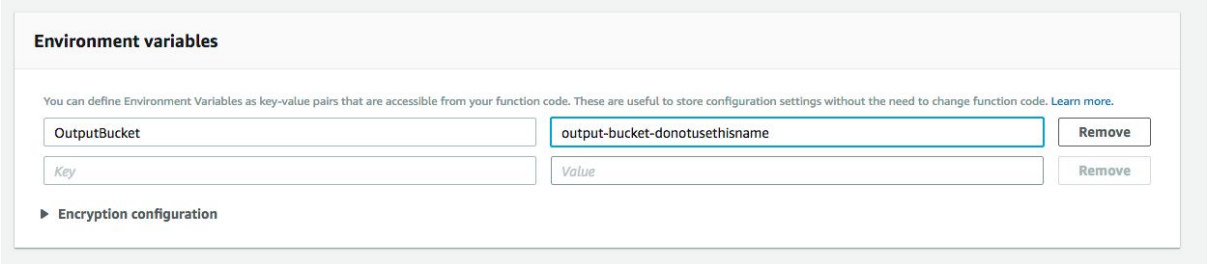
- You will now be taken to the main Lambda function page.

- Scroll down and find the code editor.

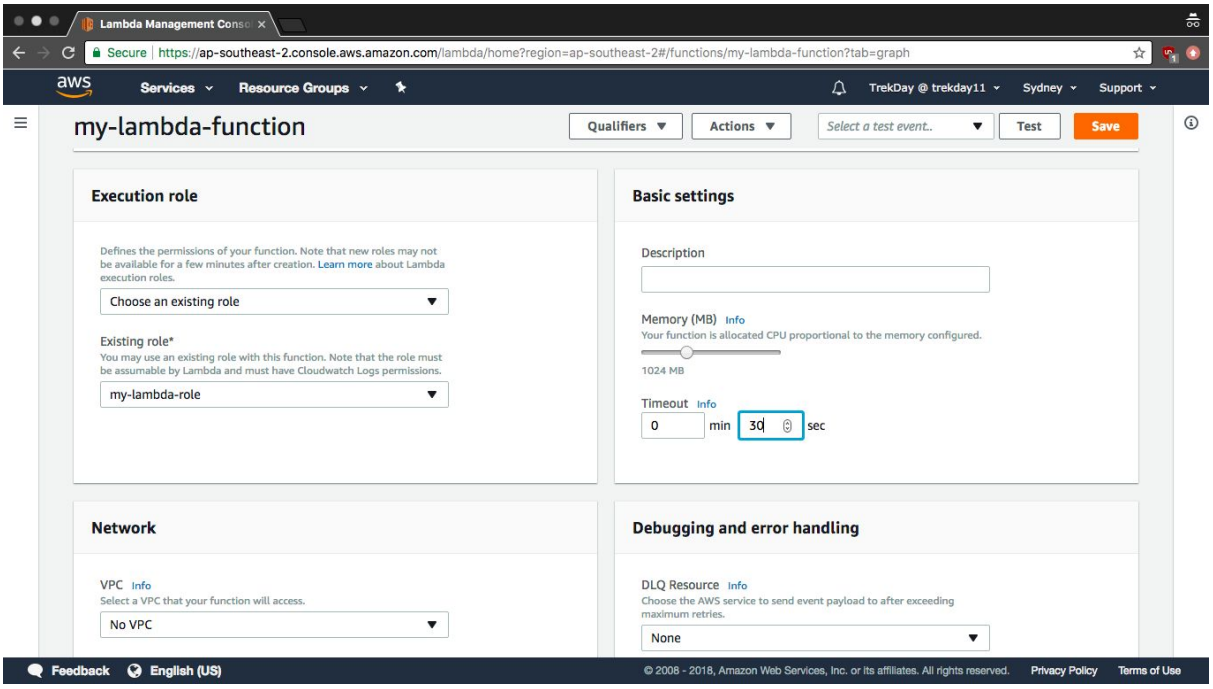
- From the resources with the session, locate 'lambda_handler.py' open up the file in a text editor. It's a pretty simple script. Copy and paste it into the Lambda code editor, replacing all the code that was there



- In the "Environment variables" section enter the "Key" of "OutputBucket" and the name of YOUR output bucket "output-bucket-xxxxxx"



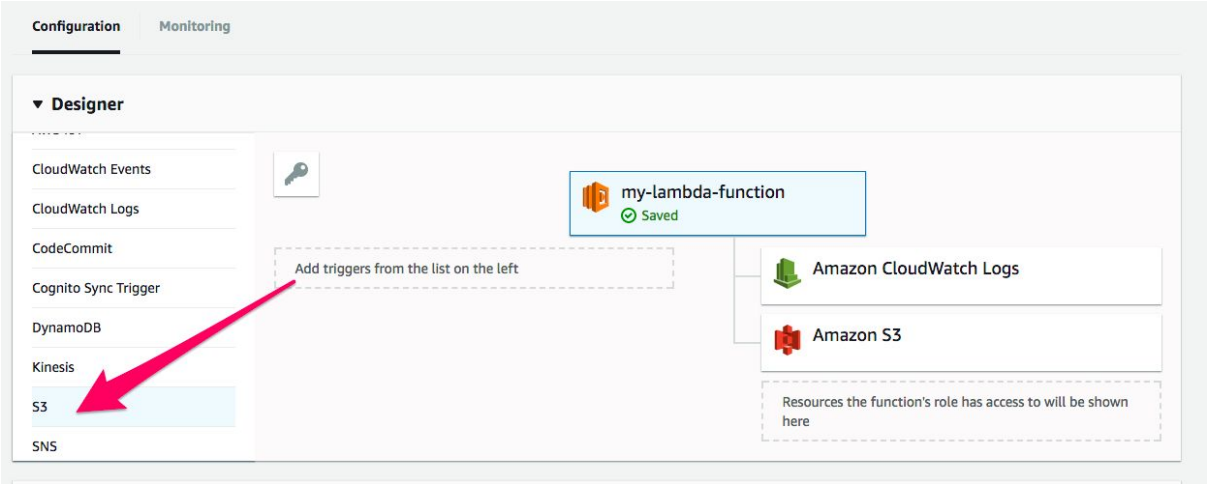
- In the "Basic settings" section move the "Memory" slider up to "1024 MB" and the "Timeout" to "30" seconds.



- Select "Save"

Adding A Trigger

- Scroll back up to the top of the page. In the "Designer" section, scroll under "Add triggers" until you see "S3" and select it.

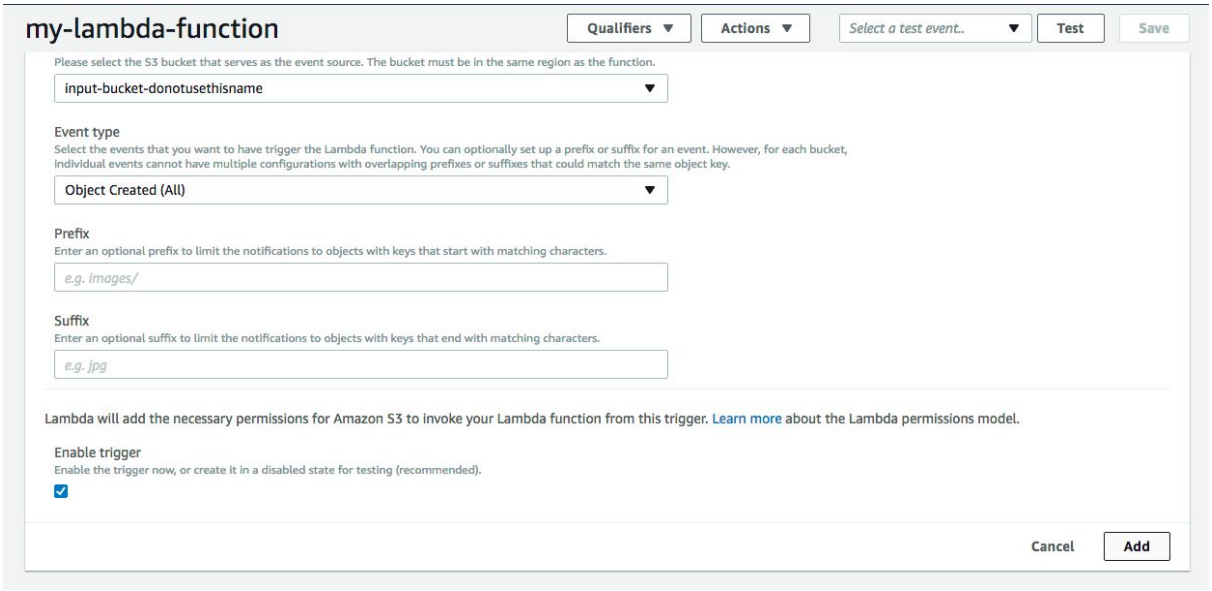


- You should now see a section called "Configure triggers"

- In the "Bucket" selection, find and select YOUR input bucket "input-bucket-xxxxxxx"

- Under "Event type" select "Object Created (All)"

- Ensure the "Enable trigger" checkbox is ticked and select "Add" (in the bottom right)

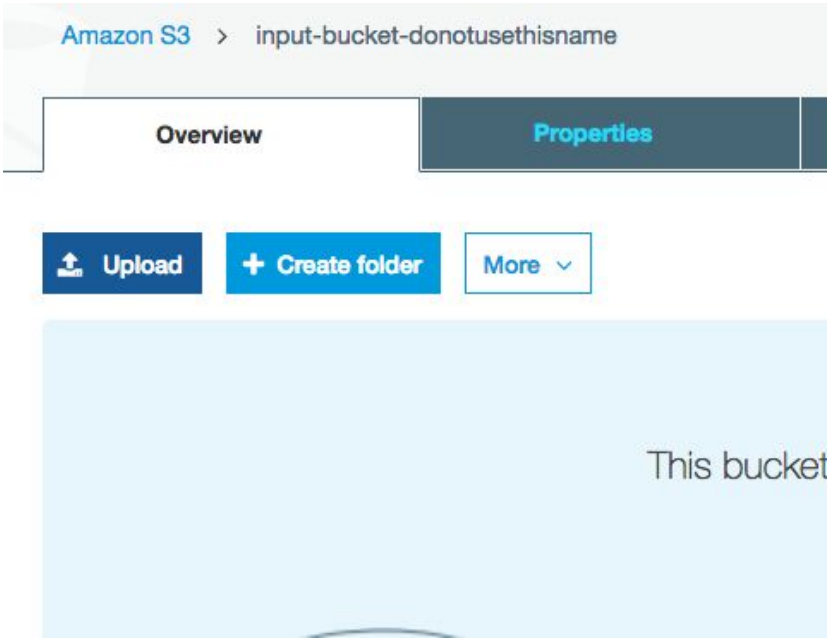


- Select "Save"

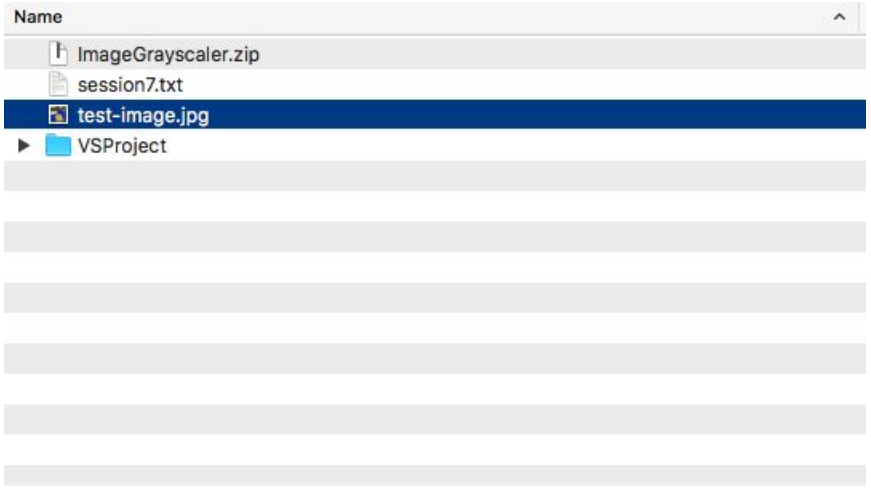
- Now it's time to test:

- Navigate to S3, and select YOUR 'input-bucket-xxxx'

- Select 'Upload'



- Within the resources for this lesson you will find a test image. Feel free to use a JPG image of your own to upload to the bucket.



Result!

Greyscale fun! :)

If you are running into issues…

- Check that your role has the appropriate rights
- Make sure you are uploading a .JPG file in your input-bucket
- Make sure that you have placed the OutputBucket Key and Value in the Environment Variables and not in Tags

Extra Challenge

While we wait for everyone to complete this session….

- Let’s get AWS to send us an email when the image has been processed!
- We will use SNS

END

Deloitte.