

Technical Excellence in the Cloud

Written for AllState, Belfast, October 2017

Contents

Chapter 1	Cloud Fundamentals
Chapter 2	Cloud Infrastructure Overview
Chapter 3	Running Servers
Chapter 4	Security in and of the Cloud
Chapter 5	Storage Options
Chapter 6	Development in the Cloud
Chapter 7	Deployment and Devops
Chapter 8	Container Services with Docker

The Cloud

Nick Todd

- [conygre] -
z i z t h g p

The Prezi presentation near the start of this training can be found here:

http://prezi.com/cxpwi_og7lht/aws-regions-and-availability-zones/

Objectives

- Introduction to the Cloud
- Benefits of the Cloud
- Strategies for Migrating to the Cloud

- [conygre] -
fill the gap

What is 'The Cloud' ?



- [conygre] -
fill the gap

Why Not Use Your Own Computer?

- Save capital expense and swap with variable expense
- Benefits of the economies of scale of the cloud provider
- Flexibility and Elasticity
- Let a company who's 'day job' is running data centres run yours so your company can focus on their 'day job' which probably isn't running data centres

- [conygre] -
fill the gap

Additional Benefits

- Many cloud providers offer sophisticated services and capabilities that would be very difficult for you to replicate
 - Elastic load balancing
 - Serverless architectures
 - Automatic replication across multiple datacentres
 - Highly available messaging
 - File storage with 11 nines of durability

- [conygre] -
fill the gap

Approaches to the Cloud

- There are several approaches that companies take to implementing the cloud
 - All in and Forklift
 - All in and Leverage
 - Hybrid Architecture
 - Backup and DR

- [conygre] -
fill the gap

All in and Forklift

- Move wholesale to the cloud
- Simply take applications you were running on premises and move them to the cloud as they are
 - Hence the term forklift
- Pros
 - Relatively easy to do
 - Relatively quick to do
 - Not much cloud knowledge required
- Cons
 - You are not benefiting fully from the capabilities of what cloud provision can achieve

- [conygre] -
fill the gap

All in and Leverage

- Take applications that are not in the cloud and rearchitect them to take advantage of cloud
- Pros
 - Benefit from all the advantages of cloud computing
- Cons
 - More complicated
 - More expensive

- [conygre] -
fill the gap

Hybrid

- Hybrid is a mix of on premises and cloud
- Some stuff cannot go to cloud
 - Regulatory compliance
 - Security requirements
- Leave this stuff on premises
- Move other stuff to the cloud
- Make sure you have a fast and secure connection between on premises and cloud provider

- [conygre] -
fill the gap

Backup and DR

- Some organisations already have data centres and contracts with data centres
- These organisations can use the cloud for backup of data
 - Remember the 11 nines of durability that some cloud providers offer
- Cloud can also be used for DR
 - Have a cold or hot standby available in the cloud in case of disaster

- [conygre] -
fill the gap

Summary

- Introduction to the Cloud
- Benefits of the Cloud
- Strategies for Migrating to the Cloud

- [conygre] -
fill the gap

Platform Overview

Platform Overview

- [conygre] -
e i d t h g a p

Objectives

- Regions
- Zones
- Servers
- Storage
- Containers
- Serverless Architectures

- [conygre] -
fill the gap

Regions

- AWS and others divide the globe up into regions
 - Each region is an implementation of the cloud platform
 - AWS regions are only connected by the Internet, whilst other cloud providers such as Google have their own infrastructure connecting regions together

- [conygre] -
fill the gap

Region Selection

- For very high availability you would deploy to multiple regions
- Region selection is based on things like
 - Latency
 - Availability of cloud features
 - Not every region will have every feature
 - Regulatory and Legal reasons
 - Copyright or data protection issues for example

- [conygre] -
fill the gap

Availability Zones

- Each region is then further divided into availability zones
- Each zone is a data centre or cluster of data centres
- Zones have very low latency high speed connections between each other
- Zones are geographically placed to minimize region wide outages

- [conygre] -
fill the gap

Running Applications

- Cloud providers allow for a number of ways to run your applications
 - Run traditional servers in the cloud
 - Run containers in the cloud
 - Run totally serverless architectures using features such as AWS Lambda

- [conygre] -
fill the gap

Running Servers – EC2

- **Elastic Compute Cloud (EC2)** servers can be launched
- Servers run in availability zones and are launched from **Amazon Machines Images (AMI)**
- AMIs can be
 - Standard AWS provided Linux and Windows variations
 - Third Party Provided – lots to choose from!
 - Created by you

- [conygre] -
fill the gap

Server Types

- When servers are launched they can be of varying sizes from
 - 0.5gb RAM with a single CPU
 - Hundreds of CPUs with almost 2000gb RAM
 - Anything in between
- Servers can be resized if required
 - Stop / Resize / Start
- Options are optimised for different tasks
 - Compute, Memory, Disk, Graphical etc.

- [conygre] -
fill the gap

Hard Drives

- EC2 servers use **Elastic Block Store** drives (**EBS**)
- EBS volumes can be
 - Magnetic
 - Solid State
 - Provisioned with a guaranteed read / write speed
- EC2 instances can have multiple EBS drives
- Drives can be moved between instances
 - Just like physical drives

- [conygre] -
fill the gap

Data Storage

- There are many storage options
 - EBS – hard drives
 - Instance store – temporary SSD storage physically attached to EC2 instances
 - ElastiCache – Memcached / Redis as a service
 - DynamoDB – NoSQL database as a service
 - RDS – relational database as a service
 - S3 – unlimited object storage
 - Glacier – archive service

- [conygre] -
fill the gap

Containers

- **Containers run as a group of namespaced processes within a single Linux operating system**
- Containers have a standard packaging format
 - Providing portability
- Containers have shorter startup times
- Containers have better resource utilization of servers
- Containers are ideal for Continuous Integration/Delivery

- [conygre] -
fill the gap

Container Services

- With the increasingly popular adoption of microservice architectures, containers are very popular with two of the main platforms
 - Docker
 - Kubernetes
- Both Docker and Kubernetes can be used on AWS for example, docker can be used with **EC2 Container Services (ECS)**

- [conygre] -
fill the gap

Serverless Architectures

- Removing the need for servers altogether is **AWS Lambda**
- AWS Lambda allows you to deploy functions to the cloud
- Functions can be triggered by things like
 - REST API calls
 - Messaging
 - File uploads
 - Scheduled events
 - Alexa voice commands

- [conygre] -
fill the gap

AWS Lambda Benefits

- You only pay in 100ms chunks for the time the function is actually running
- There is no infrastructure to support
- You can specify the amount of memory the lambda runs with
- Lambdas can be written in
 - Java
 - JavaScript
 - Python
 - C#

- [conygre] -
fill the gap

Summary

- Regions
- Zones
- Servers
- Storage
- Containers
- Serverless Architectures

- [conygre] -
fill the gap

Running Servers

Running Servers

- [conygre] -
e i t z t h * g * p

Objectives

- Setting up servers
- Setting up the network
- Server security

- [conygre] -
fill the gap

Running Servers

Running Servers

- The following key terms are important to understanding how to run servers in AWS
 - Amazon Machine Images
 - EC2 instances
 - Virtual Private Clouds
 - Instance types

- [conygre] -
fill the gap

Amazon Machine Images



- An Amazon Machine Image (AMI) is the baseline install for your server
- AMIs can be
 - Windows or Linux
 - Provided by Amazon, you, or a third party
 - Can include your entire application, or just be a basic Windows / Linux operating system
- An AMI is the template for your server
 - Like a VMWare Appliance or a Virtual Box image

- [conygre] -
fill the gap

Running Servers

EC2 Instances

The diagram illustrates the creation process of an EC2 instance. It starts with an 'AMI' (Amazon Machine Image) icon, which is connected by an arrow labeled 'AMI used to create' to an 'EC2 Instance' icon. This instance icon is then connected by an arrow labeled 'EC2 has a network attached drive' to an 'EBS Volume' icon.

```
graph LR; AMI[AMI] -- "AMI used to create" --> EC2[EC2 Instance]; EC2 -- "EC2 has a network attached drive" --> EBS[EBS Volume]
```

- [conygre] -
fill the gap

Running Servers

EC2 Instance Sizing

- EC2 Instances can be of different sizes

Range	Role
T / M	General Purpose servers
C	Compute optimised
I	IO Optimised
G	Graphical processing optimised
R	Memory optimised

- For a full description visit
 - <https://aws.amazon.com/ec2/instance-types/>
- You can change an instance type by stopping / resizing / restarting a server

- [conygre] -
fill the gap

Block Storage



- The standard block storage option is Elastic Block Store (EBS)
- EBS volumes are network attached and are used by EC2 instances as their hard drives
- EBS volumes can have the following options
 - Magnetic
 - Solid State (the default)
 - Solid State Provisioned (provisioned level of disk IO)
 - Solid State Network Optimized (improved Network IO)
- EBS volumes can also be encrypted

- [conygre] -
fill the gap

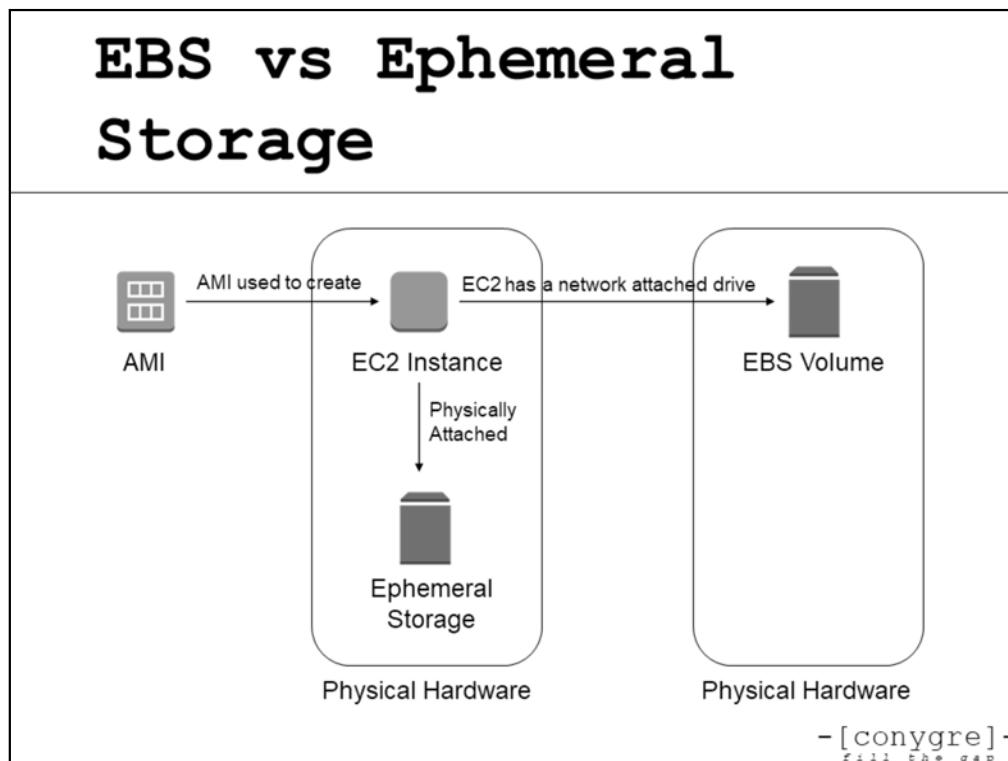
Ephemeral Storage



- EC2 instances can also have physically attached drives referred to as ephemeral storage and sometimes referred to as instance storage
- When an EC2 instance is stopped and started they always restart on different hardware
 - Therefore, instance store drives do not survive an EC2 restart
- It is also possible to have EC2 instances that run exclusively on instance store
 - These cannot be stopped and started, only terminated

- [conygre] -
fill the gap

Running Servers



Running Servers

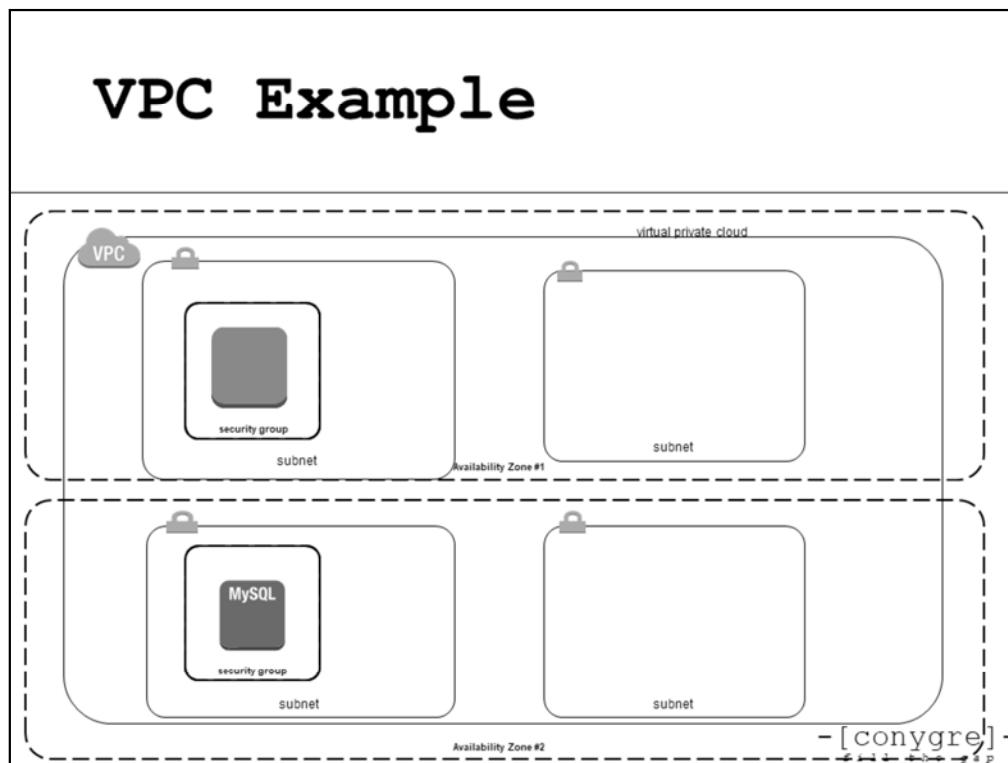
Networking



- EC2 instances run within a Virtual Private Cloud (VPC)
- The VPC is the virtual version of a physical networking environment consisting of
 - IP range
 - Subnets within that range
 - Internet Gateways, and NAT servers
 - VPN Gateways
- A VPC spans availability zones within a region so you can easily run your applications across multiple facilities

- [conygre] -
fill the gap

Running Servers



VPC IP Range

- A VPC has an IP address range
- VPCs can be connected together
 - Only if the IP ranges do not overlap
- VPCs can be connected to on premises via
 - VPN Gateway
 - AWS Direct Connect



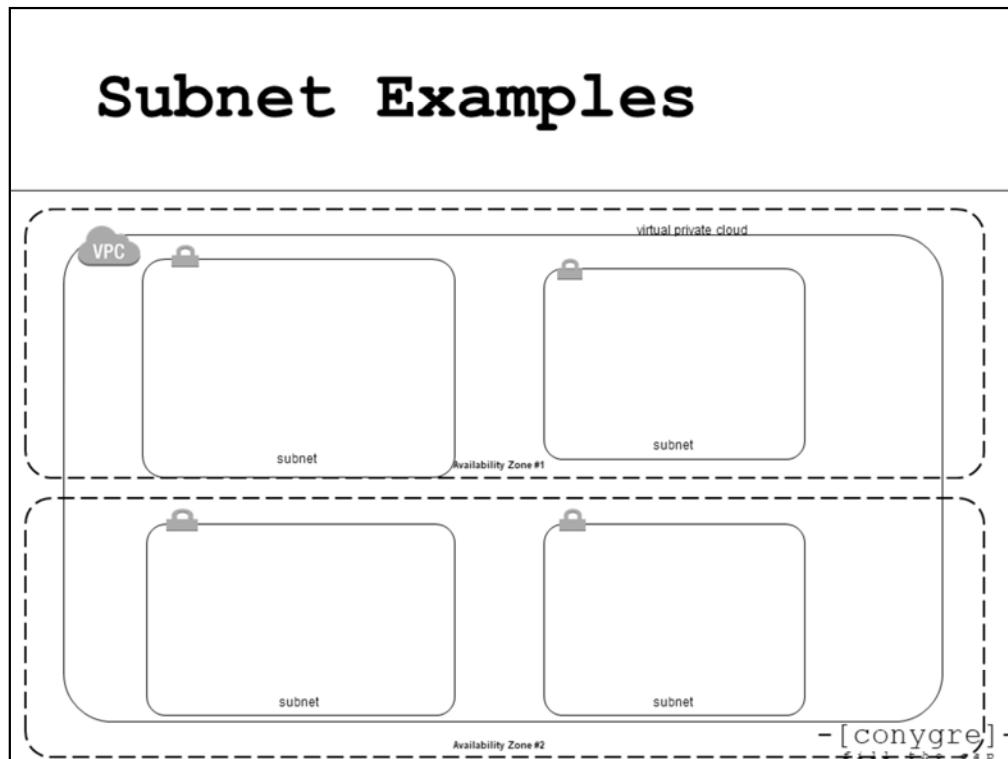
- [conygre] -
fill the gap

Subnets

- A VPC is broken down into **Subnets**
- A Subnet is a subset of the IP addresses in the VPC
- A Subnet
 - Cannot span availability zones
 - Can be on or off the Internet controlled through routing rules
- Subnets to be available to the Internet route through an **Internet Gateway**

- [conygre] -
fill the gap

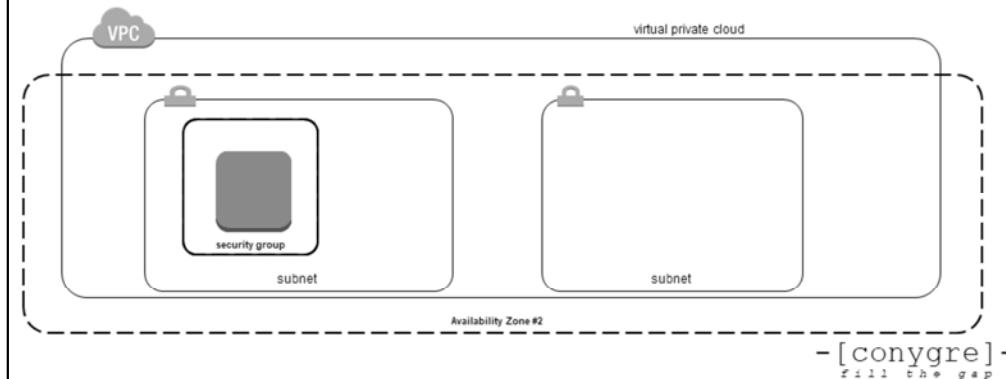
Running Servers



Running Servers

Security Groups

- EC2 instances are launched into subnets, and the access to ports is controlled by **security groups**



Running Servers

Security Group Examples

- Web Servers

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0

- Remote Desktop from Anywhere

Type	Protocol	Port Range	Source
RDP	TCP	3389	0.0.0.0/0

- SSH from a specific address

Type	Protocol	Port Range	Source
SSH	TCP	22	77.101.210.209/32

- [conygre] -
fill the gap

Server LifeCycle

- Servers can be
 - Stopped
 - Restarted
 - Terminated
- Restarting a server causes it to relaunch on new physical hardware in case you are restarting due to a hardware fault
 - Be aware this means you will lose your ephemeral storage!

- [conygre] -
fill the gap

Summary

- Setting up servers
- Setting up the network
- Server security

- [conygre] -
fill the gap

Exercises

- Introduction to EC2
 - <https://qwiklabs.com/focuses/3459>
- Introduction to VPC
 - <https://www.qwiklabs.com/focuses/7317>

- [conygre] -
fill the gap

Security

Cloud Security

- [conygre] -
c o n y g r e
c l o u d s h a p e

Objectives

- Security concerns
- Securing your Account
- Security your infrastructure
- Security your data
- Encryption
- Auditing tools

- [conygre] -
fill the gap

Security Concerns

- Of all the reasons people give for not moving to the cloud, probably the most common one given is security
 - Sometimes this can be more out of ignorance than fact
- However, clearly security is of vital importance, for example
 - Securing accounts
 - Securing servers
 - Securing data

- [conygre] -
fill the gap

Securing Accounts



- When you set up an AWS account, you will have a username and password
 - This should **NEVER** be used
- Instead of using your root account, AWS provides
 - **Identity and Access Management (IAM)**

- [conygre] -
fill the gap

Understanding IAM



- IAM allows you to create
 - Users
 - Groups
 - Roles
 - Policies



Users



Groups



Policies



Roles

- [conygre] -
fill the gap

Users



- **Users** are created for those individuals who need some kind of access to your AWS infrastructure
- They are for things like
 - Creating resources on AWS
 - Viewing resources on AWS
- They are NOT for
 - Database accounts
 - Operating system accounts
 - Application accounts

- [conygre] -
fill the gap

Groups



- **Groups** are as the name implies, groups of users
- Groups might be
 - Devs
 - Testers
 - Infrastructure Engineers
 - DevOps
 - SysOps

- [conygre] -
fill the gap

Roles



- **Roles** are used as
 - Temporary permissions for individuals
 - Temporary permissions for applications
 - Eg. A mobile app requiring permission to upload to S3
 - Services in AWS to have permission to do things
 - An EC2 instance to listen to a queue
 - A Lambda function to be allowed to stop and start servers

- [conygre] -
fill the gap

Policies and Permissions

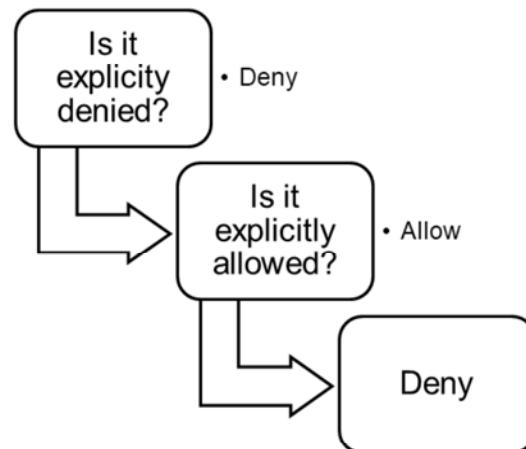


- Users, Groups and Roles can be assigned permissions in the form of **Policies**
- Policies are used to grant or deny permission to AWS resources and services

- [conygre] -
fill the gap

Default Permissions

- Everything is denied by default unless explicitly allowed
- Anything explicitly denied can never be allowed



- [conygre] -
fill the gap

Policies

- Policies can be
 - Custom policies created by you
 - Hand crafted with extremely fine grained access defined
 - AWS Provided
 - Simple policies you can use when required
 - EC2 Full Access
 - S3 Full Access

- [conygre] -
fill the gap

Securing Servers - YOU

- **YOU** are responsible for
 - Patching and updating your servers
 - Malware protection and antivirus
 - Ensuring that you only open the appropriate ports through security groups
 - Ensuring that you place them in appropriate subnets with the correct routing rules

- [conygre] -
fill the gap

Securing Servers - AWS

- Amazon are responsible for ensuring
 - Ensuring that all your security groups and routing rules work as you have specified
 - EC2 instances are completely isolated from each other
 - The security of the physical data centres
 - That all disks at end of life are destroyed to the appropriate international standards

- [conygre] -
fill the gap

Securing Data

- You are responsible for things like
 - For putting the appropriate permissions and access rights onto your S3 buckets, DynamoDB tables, RDS databases etc.
- AWS are responsible for things like
 - Ensuring the permissions you have set actually work
 - Backups and patching of RDS instances happens according to your schedule

- [conygre] -
fill the gap

Encrypting Data

- Many AWS data services support encryption for example, S3 buckets and EC2 drives can be encrypted at rest with keys
 - Provided by AWS
 - Provided by you
- AWS even provide a hardware based key management solution that you can take advantage of at extra cost

- [conygre] -
fill the gap

CloudTrail



- CloudTrail tracks all your IAM user and root user actions within your account
 - Who does what and when
 - Who stopped that server?
 - Who created that queue?
- You need to enable it within your account
- All data goes into an S3 bucket and can be viewed via the console or queried through the CLI
- Great for audit trails

- [conygre] -
fill the gap

AWS Config

- AWS Config allows you to track configuration changes to your resources over time
 - How has this server been changed over time?
 - Resized
 - Stopped / Started
 - Changed security groups
- Also great for audit trails

- [conygre] -
fill the gap

Trusted Adviser

- AWS also has a trusted adviser which will review all of your resources and highlight potential security issues

- [conygre] -
fill the gap

Security Questions

- The AWS Five Pillars also highlight a number of questions you can use to critically look at your security

- [conygre] -
fill the gap

Exercise

- Introduction to Identity and Access Management
 - <https://qwiklabs.com/focuses/6211>

- [conygre] -
fill the gap

Summary

- Security concerns
- Securing your Account
- Security your infrastructure
- Security your data
- Encryption
- Auditing tools

- [conygre] -
fill the gap

Storage Options

Data Options on AWS

- [conygre] -
e i i t h g a p

Storage Options

Objectives

- Overview of the data storage options
- S3
- Glacier
- Relational Data Store (RDS)
- Redshift
- Dynamo DB

- [conygre] -
fill the gap

Storage Options

Overview

- There are multiple data options on AWS
- Relational Database options
 - RDS
 - Redshift (big data)
- NoSQL options
 - Dynamo DB
 - ElastiCache
- Object Storage
 - S3
 - Glacier
- Disk storage
 - Elastic Block Store
 - Elastic File Store
 - Instance Store

- [conygre] -
fill the gap

Storage Options

Relational Data Store (RDS)



- RDS is where AWS provide the database instance for you so you don't have to deal with
 - Installation
 - Patching
 - Backups
 - Master / Slave configuration
 - Read Replica configuration (on some databases)
 - Complexities around resizing a live database instance

- [conygre] -
fill the gap

Storage Options

RDS Supported Databases

- Oracle
- SQL Server
- MySQL
- Aurora (big data version of MySQL)
- Postgres



- [conygre] -
fill the gap

Storage Options

RDS Considerations

- When using RDS you can choose
 - Different instance sizes
 - Backup time
 - Patching time
 - Availability Zone locations for Master/Slave
 - Encryption at rest and or in transit
 - Access is controlled using Security Groups
(restricts ports and IP ranges)

- [conygre] -
fill the gap

Storage Options

RDS Example

Specify DB Details

Instance Specifications

DB Engine	mysql
License Model	general-public-license
DB Engine Version	5.6.27

Review the Known Issues/Limitations to learn about potential compatibility issues with specific database versions.

DB Instance Class	db.m3.xlarge — 4 vCPU, 15 GiB RAM
Multi-AZ Deployment	Yes
Storage Type	Provisioned IOPS (SSD)
Allocated Storage*	100 GB
Provisioned IOPS	1000

- [conygre] -
fill the gap

Storage Options

Big Data with Redshift

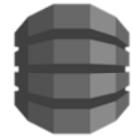


- According to the AWS Web site
 - *“Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse solution that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools.”*
- Redshift can be used for massive amounts of data
- Redshift can be queried using SQL to gain information from your data

- [conygre] -
fill the gap

Storage Options

NoSQL with DynamoDB

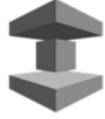


- For NoSQL data, AWS provides DynamoDB
- According to AWS
 - *“Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale”*
- You pay for the provisioning of throughput and the storage
- DynamoDB supports key based and document based storage

- [conygre] -
fill the gap

Storage Options

NoSQL with ElastiCache



- ElastiCache is a caching technology that supports both Memcached and Redis caches
- ElastiCache is an extremely fast in memory cache that can also be used for temporary data storage



- [conygre] -
fill the gap

Storage Options

Object Storage with S3



- S3 provides object storage for files of up to 5TB in size
- The storage is unlimited and has 11 nines of durability with your files saved on multiple drives across multiple data centres
 - *If you store 10,000 objects with us, on average we may lose one of them every 10 million years or so. This storage is designed in such a way that we can sustain the concurrent loss of data in two separate storage facilities.*

- [conygre] -
fill the gap

Storage Options

S3 Buckets



- When using S3 you create buckets into which you place objects
- These objects then have a key which is what is used to retrieve the object
- Buckets can be made available over the Internet
 - Static Web sites can be hosted from buckets
 - CSS / JS / Images can be hosted from buckets

- [conygre] -
fill the gap

Storage Options

Glacier



- Glacier is the data archiving service
- It is ideal for data that you have to store but are unlikely to want to access
 - Historical logs
 - Archives
 - Old Backups
 - Regulatory compliance
- Glacier is similar to S3 with 11 9's of durability
- S3 objects can be put into a lifecycle automating a transition to Glacier

- [conygre] -
fill the gap

Storage Options

Exercises

- Introduction to the Relational Database Service
 - <https://qwiklabs.com/focuses/3463>
- Introduction to S3
 - <https://www.qwiklabs.com/focuses/2355>

- [conygre] -
fill the gap

Storage Options

Summary

- Overview of the data storage options
- S3
- Glacier
- Relational Data Store (RDS)
- Redshift
- Dynamo DB

- [conygre] -
fill the gap

Developing for the Cloud

- [conygre] -
f i d t h g a p

Objectives

- General best practices
- Statelessness and Scaling
- Monitoring
- Loose coupling
- SDKs and APIs
- API Examples

- [conygre] -
fill the gap

General Best Practices

- Cloud based architectures ***significantly change*** how you approach development
- Key concepts are
 - Stateless servers
 - Decoupled infrastructure
 - Horizontal scaling wherever possible
- To illustrate we will use the example of a standard shopping Web site

- [conygre] -
fill the gap

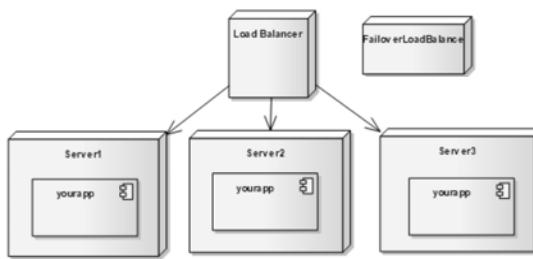
Standard Web Site

- Many Web sites store session state
 - Where do you keep it?
- If you keep it on the server, this prevents elasticity
 - Servers cannot be stopped and started automatically in response to load
 - Stopping servers will result in the loss of session state
- **Cloud based servers should be stateless wherever possible**

- [conygre] -
fill the gap

Horizontal / Vertical Scaling

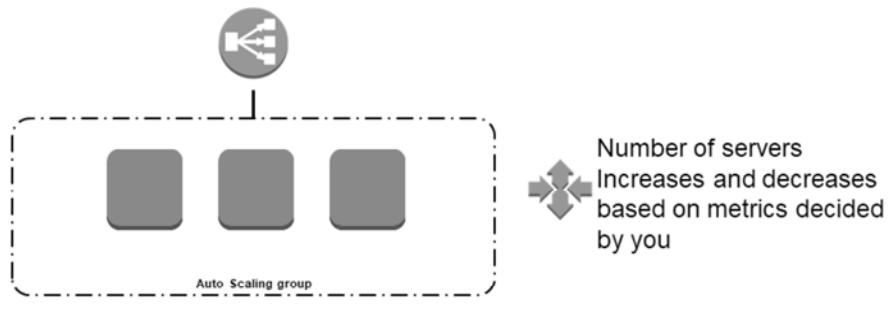
- Vertical scaling always has limits
- Horizontal scaling is far less limiting
 - Will your application horizontally scale?
- Building components that cannot be horizontally scaled lacks flexibility in the cloud

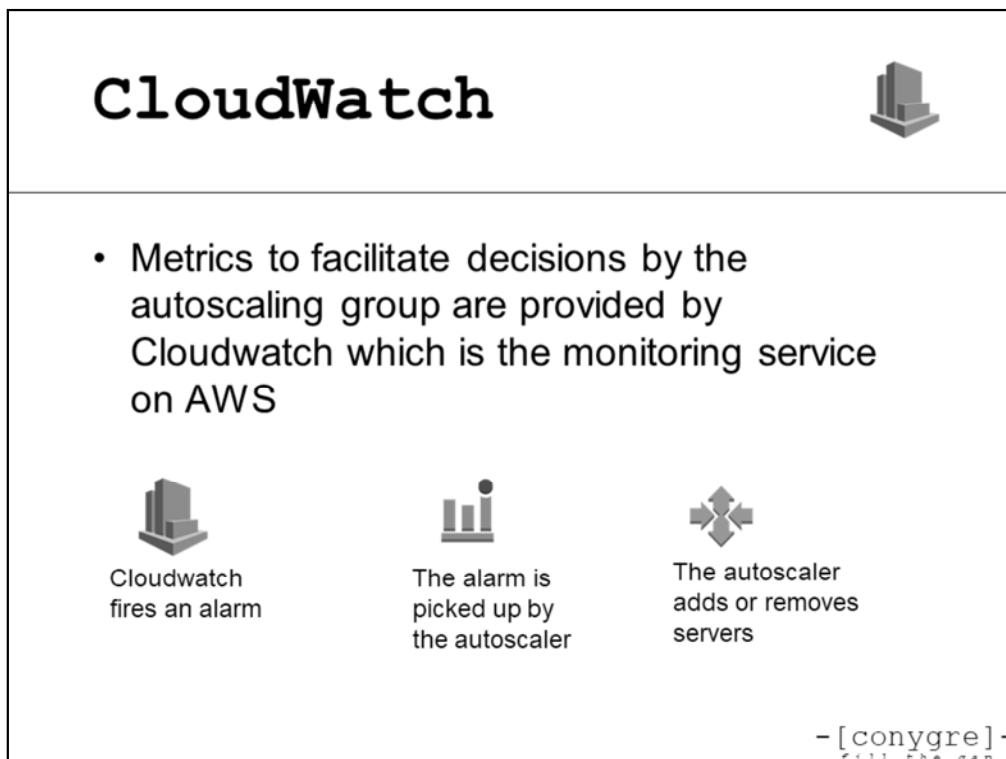


- [conygre] -
- z i l l t h e g a p -

Horizontal Scaling on AWS

- The Elastic Load Balancer and the AutoScaling Group facilitate horizontal scaling

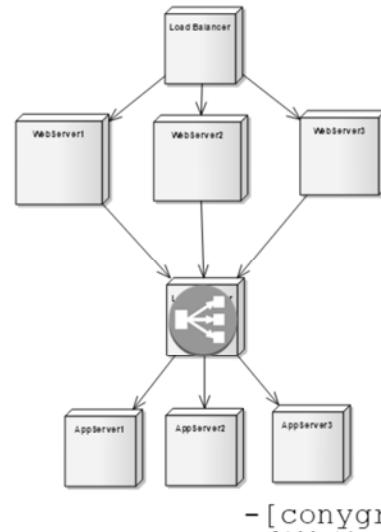




Loose Coupling



- Tightly coupled servers are unnecessarily inflexible
- Direct servers to load balancers NOT server IP addresses



- [conygre] -
t i l l t h e g a p

Development in the Cloud

Messaging

- Loose coupling through messaging is also effective
- AWS provides the **Simple Queue Service** to facilitate messaging

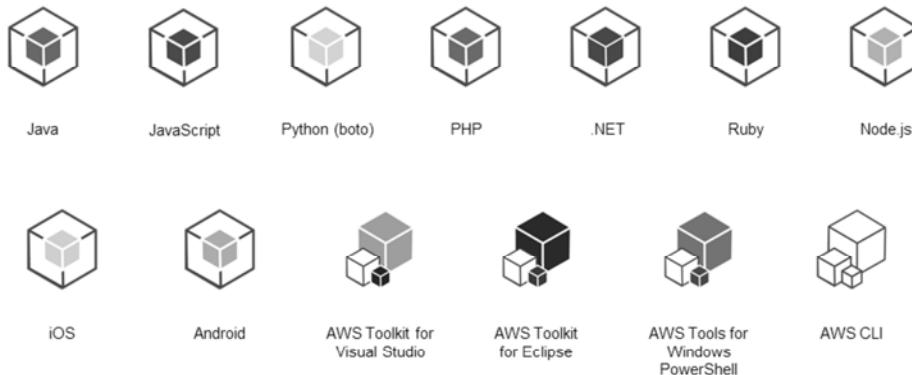
```
graph TD; LB[Load Balancer] --> WS1[WebServer1]; LB --> WS2[WebServer2]; LB --> WS3[WebServer3]; WS1 --> MQ((Message Queue)); WS2 --> MQ; WS3 --> MQ; MQ --> AS1[AppServer1]; MQ --> AS2[AppServer2]; MQ --> AS3[AppServer3]
```

- [conygre] -
fill the gap

Development in the Cloud

Developing for the Cloud

- AWS Provides SDKs for the cloud



- [conygre] -
fill the gap

Application APIs

- AWS provide APIs for various languages and platforms
- APIs exist for all of the services allowing you to implement features into your applications

- [conygre] -
fill the gap

Example S3 Code to List Bucket Contents

- Below is an S3 code snippet in Java

```
BasicAWSCredentials awsCreds = new BasicAWSCredentials(accessKey, secretKey);
s3client = new AmazonS3Client(awsCreds);
List<String> keyNames = new ArrayList<String>();
System.out.println("Listing objects");
ListObjectsRequest listObjectsRequest = new ListObjectsRequest()
ObjectListing objectListing;
do {
    objectListing = s3client.listObjects(listObjectsRequest);
    for (S3ObjectSummary objectSummary : objectListing.getObjectSummaries()){
        String keyName = objectSummary.getKey();
        System.out.println(" - " + keyName + " " + "(size = "
            + objectSummary.getSize() + ")");
        if (!keyName.endsWith("/")){
            keyNames.add(keyName);
        }
    }
    listObjectsRequest.setMarker(objectListing.getNextMarker());
}
```

- [conygre] -
fill the gap

Example Lambda for Stopping Instances

- Below is a Lambda Function in Python

```
import boto3
import logging
ec2 = boto3.resource('ec2')

def lambda_handler(event, context):
    filters = [
        {
            'Name': 'tag:Role',
            'Values': ['MyRole']
        },
        {
            'Name': 'instance-state-name',
            'Values': ['stopped']
        }
    ]
    instances = ec2.instances.filter(Filters=filters)
    StoppedInstances = [instance.id for instance in instances]
    if len(StoppedInstances) > 0:
        startingUp = ec2.instances.filter(InstanceIds=StoppedInstances).start()
```

- [conygre] -
fill the gap

Exercise

- Introduction to AWS Lambda
 - <https://qwiklabs.com/focuses/2966>

- [conygre] -
fill the gap

Summary

- General best practices
- Statelessness and Scaling
- Monitoring
- Loose coupling
- SDKs and APIs
- API Examples

- [conygre] -
fill the gap

Deployment and Devops

DevOps and Deployment

- [conygre] -
f i d t h g a p

Objectives

- Benefits of the Cloud for Deployment
- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- CloudFormation
- UserData
- OpsWorks

- [conygre] -
fill the gap

Deployment

- Deployments to the cloud enable all sorts of capabilities you would otherwise not have
 - Deployment can include the complete creation of a virtual environment
 - Servers don't need to have applications redeployed
 - You simple create new servers for every deployment
- Infrastructure can be in source control
- You can easily create test environments

- [conygre] -
fill the gap

Continuous Integration

- This is largely unaffected by the cloud
- You can use your favorite CI server to continually build and test your code when checked in
- If desired, you can use **AWS Code Commit** as your source code repository

- [conygre] -
fill the gap

Continuous Delivery

- Continuous Delivery can be achieved easily by your CI server or CD server
- What changes in the cloud are two key things
 - Where do your deliveries end up?
 - What makes up a delivery?

- [conygre] -
fill the gap

CD Location

- Continuous delivery can place artifacts into S3 ready to be retrieved by your deployment pipeline
- S3 makes this straightforward since you can place appropriate IAM permissions around the bucket

- [conygre] -
fill the gap

CD Artifact

- Typically, a CD artefact would be a
 - war file
 - .Net project
 - NodeJS project
 - Web site
- If you are using containers, then the artefact could be a
 - Docker image

- [conygre] -
fill the gap

AMIs as Artifacts

- When using AWS, you can also build an AMI as part of your CD process
- The AMI contains all of the software and configurations of the server so it is ready to launch
 - Resulting in a fast launch time
- Netflix outsourced a utility to help with this
 - **Aminator**

- [conygre] -
fill the gap

Continuous Deployment



- How could you go about continuous deployment?
 - This is where it can get interesting in the cloud
- Using **CloudFormation** you can script the creation of your entire infrastructure
 - Infrastructure as code

- [conygre] -
fill the gap

Infrastructure Creation

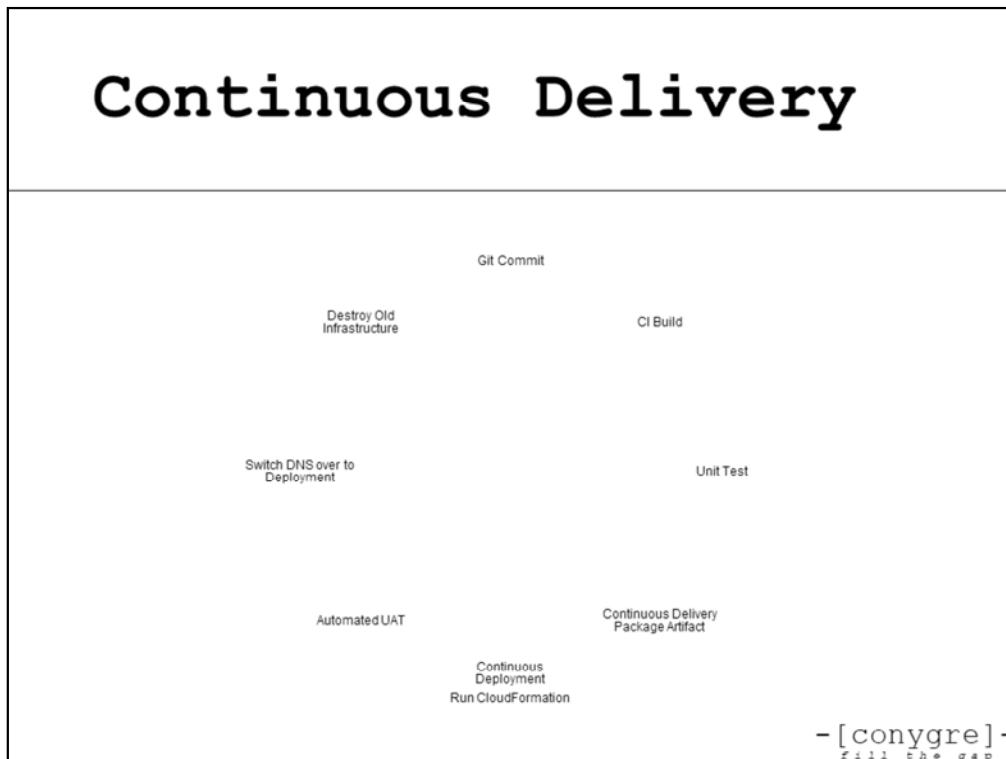
- To automate the creation of infrastructure, AWS provide **Cloudformation**
- Cloudformation allows you to maintain ***infrastructure as code***
- JSON files are maintained in version control that define all the resources you need to deploy
 - This can then be run as a script whenever you need to

- [conygre] -
fill the gap

CloudFormation Templates

- A CloudFormation Template can be used to
 - Create servers
 - Create network infrastructure
 - Create autoscaling groups
 - Create IAM users
- This facilitates Continuous Delivery with a new infrastructure every time

- [conygre] -
fill the gap



Manage the Pipeline

- The pipeline for continuous delivery can all be done using Jenkins
 - One job is configured to complete the builds
 - Another job is configured to pick up the deployments and push to production via CloudFormation

- [conygre] -
fill the gap

Exercise

- Introduction to CloudFormation Designer
 - <https://qwiklabs.com/focuses/3468>

- [conygre] -
fill the gap

Summary

- What do we mean by ‘the cloud’?
- Benefits of the Cloud
- Corporate Approaches to the Cloud
- How does the Cloud affect Development

- [conygre] -
fill the gap

Container Services with Docker

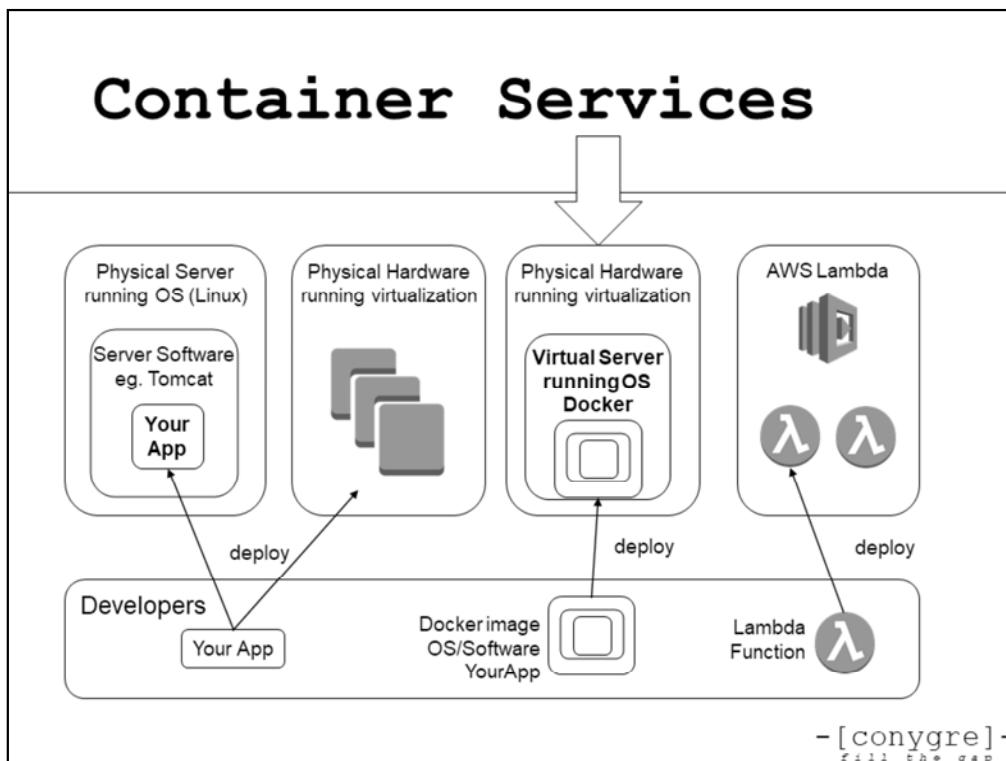
- [conygre] -
f i d t h g a p

Objectives

- What is Docker
- Installing Docker
- Creating an Image
- Running an Image
- Pushing an Image to a Repository

- [conygre] -
fill the gap

Docker



Container Services

- Containers services such as docker allow developers to create deployments that consist of
 - OS / Required Software
 - Application all configured and ready to run
- These deployments are called **Docker Images**
- The **Docker Container** can then be used to run your images in production
 - You are deploying a complete OS + Software created and tested by the developers

- [conygre] -
fill the gap

Getting Started

- The first thing you will need to do in order to create and run an image is to install Docker
- The process is fairly straightforward, for example

```
sudo yum install -y docker  
sudo service docker start
```

- [conygre] -
fill the gap

Creating an Image

- Docker images are created using a config file called **Dockerfile**
- The Dockerfile is a manifest that specifies
 - The base image you wish to use for your instance
 - What software should be installed
 - What files should be available on it

- [conygre] -
fill the gap

Dockerfile Syntax

- The first line in a Dockerfile specifies the base image using the keyword **FROM**

```
FROM ubuntu
```

- You can specify the author using **MAINTAINER**

```
MAINTAINER Nick
```

- Commands to run at build can be run with **RUN**

```
RUN apt-get update -y  
RUN apt-get install -y apache2
```

- [conygre] -
fill the gap

Specifying Service to Run – CMD

- The command to be used to run the software on the image once built is **CMD**

```
CMD ["executable", "param1", "param2"...]
```

- For example, to launch Apache Web server

```
CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```

- CMD is typically used when you might want to override the command that is used when you launch the container

- [conygre] -
fill the gap

Specifying Service to Run - ENTRYPOINT

- The **ENTRYPOINT** is similar to the CMD in that it specifies what happens when the image launches
 - It is not so easy to override as CMD
 - ENTRYPOINT is normally used for images that are running server software where you are unlikely to want to override it
- Any parameters passed to CMD will be passed to ENTRYPOINT instead

```
CMD "Hello World!"  
ENTRYPOINT echo
```

- [conygre] -
fill the gap

Other Dockerfile Keywords

Keyword	Role
LABEL	Adds meta data. Each LABEL creates a new layer, so don't have too many!
EXPOSE	Which ports to open from the image
COPY	Copy files into the image
VOLUME	Mount volumes on the host into the image
USER	Specify the user that the image command runs as

- [conygre] -
fill the gap

Dockerfile Example

- Launching Apache Web server

```
FROM ubuntu:12.04
# Install dependencies
RUN apt-get update -y
RUN apt-get install -y apache2
# Install apache and write hello world message
RUN echo "Hello World!" > /var/www/index.html
# Configure apache
RUN a2enmod rewrite
RUN chown -R www-data:www-data /var/www
ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
EXPOSE 80
CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```

-[conygre] -
fill the gap

Building an Image

- Once you have the Dockerfile you can build the image
 - `docker build -t hello-world .`
- The `-t` sets the name of the build and the `.` signifies the location of the dockerbuild file

- [conygre] -
fill the gap

Viewing your Images

- You can view all of your images using the **docker images** command

```
nicktodd$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED
mod1               latest   2b50d6da7c91    8 weeks ago
338MB
<none>              <none>   bf8269f59568    8 weeks ago
340MB
nbrown/revealjs    onbuild  7712702d2a15    2 months ago
334MB
myweatherapp       latest   5f45f7549286    2 months ago
911MB
weather-microservice latest   c8b022f24aa9    4 months ago
911MB
microsoft/dotnet   1.1-sdk-msbuild a323f2a05797    7 months ago
878MB
hello-world        latest   690ed74de00f    23 months ago
960B
960B
NickMac:~ nicktodd$
```

- [conygre] -
fill the gap

Running the Image

- To run the image use **docker run**

```
docker run -p 80:80 hello-world
```

- The **-p** sets the port number in the image and what port it should forward to on the host
- You can then test this image in the browser by visiting **http://localhost**

- [conygre] -
fill the gap

EC2 Container Registry



- Docker images can be registered in the **Amazon EC2 Container Registry**
 - A managed container registry service
- Docker provides a CLI for pushing images and managing images in the registry
- A registry can be created through the AWS CLI or the AWS Console

- [conygre] -
fill the gap

Exercise

- Introduction to Amazon EC2 Container Registry
 - <https://qwiklabs.com/focuses/3456>

- [conygre] -
fill the gap

Summary

- What is Docker
- Installing Docker
- Creating an Image
- Running an Image
- Pushing an Image to a Repository

- [conygre] -
fill the gap