AWS – Getting Started

Developing and Deploying to AWS

* Deploying scalable applications (Elastic Beanstalk)
* Storing Static Content
* Creating Reproducible Infrastructure with Cloud Formation Templates

AWS – The Big Picture

* Core Services of AWS
* Elastic Cloud Compute
* Virtual Machines
* Extended Services
* Different ways to access AWS

**Core Services – Elastic Cloud Compute (EC2), Simple Storage Service (S3), Relational Database Service (RDS), Route53**

Harnessing the Power of AWS from the Command Line to Code –

* Web Console
* Command Line Interface Utility
* Software Development Kits (SDK)

AWS Service Health Dashboard

<http://status.aws.amazon.com>

\*Most AWS Bugs are misconfigured Security Groups

**Elastic Cloud Compute (EC2)**

**An Instance** can be used to Run Applications, run Virtual Desktops/Machines, Run 3rd Party Software, and Computing. It can basically do anything, and is a computer.

Computing services operating in remote data servers around the world. **Elastic refers to** the computing service being able to expand and retract as needed.

The basic building block in EC2 is an instance. An instance is basically a virtual server

When Creating an instance, the first is selecting an Amazon Machine Image (AMI)

**Amazon Machine Image (AMI)** is an Operating System + Software used on an EC2 Instance

Amazon updates the Image Software … Not my instance… So, it will stay updated until I create it. Once this occurs, I will need to manually update or create a new instance and migrate the existing code (I need to do research to see what is common and the decision making behind what is the preferred path \*\*\*\*\*\*\*)

Once you have selected an Image for your Instance, the next thing you will do is to select your Instance Type. This is basically the specs for your Instance, cpu, ram, etc.

* You can create families or sub-categories for your Instance Types.

Large Instance Type Comparison – Its important to know what you will need in regard to vCPU and Memory. The price is changed based off the size.

Configuring the Instance Details is the next step and helps with scaling.

Adding Storage allows you to calculate Storage and you decide by adding Elastic Block Storage. It is specifically for using EC2. You can also add Volumes and adjust the Size needed for each.

Step 5 is for tags which are not that important right now.

At step 6 we have the Security Group**. The Security Group** is like a little firewall or IP-based communication rules for a single or group of service instances.

* An example of Security Group scenarios would be – Controlling who can SSH into EC2 Instance
* Allow access between EC2 Instances
* Allow access to Databases
* Accept HTTP Requests

Finally review the instance in step 7 and create the instance with an existing key pair. This allows us to SSH into the instance and make whatever modifications we want.

EC2 Instances are charged by the hour. Instance Type, AMI Type change the price.

**Simple Storage Service (S3)**

Widely used as the place to store files. Maximum file size of 5 terabytes.

Buckets are the foundational structure of S3. Main resource of add, modifying and deleting objects.

Buckets can

* trigger events when objects are added/modified/deleted
* preserve older versions of objects
* replicate objects across regions

Buckets are accessed via URLS. Example below:

<https://s3-us-west-1.amazonaws.com/okfido.org/img/okfido_logo.png>

s3-us-west-1 is the S3 Bucket Region

okfido.org is the Bucket Name

okfido\_logo.png is the Object Path

**Route53 – DNS management**

Allows you to easily configure domain names. DNS Stands for Domain Name System – A system that translates human-readable URLs to IP addresses.

Certain amazon services like S3 Buckets or Load Balancers don’t have static visible IP addresses. Route53 allows you to set up URL resolutions to AWS services directly, bypassing the need to see an IP.

Route53 is Core to letting users interact with services in AWS.

**Elastic Cloud Compute (EC2)** – creating instances you are able to run 3rd party apps or your own code on a scalable machine in the cloud.

**S3** – Uses are endless, great place to store

**RDS for databases** – not worrying about right now

**Route53 –** how to expose AWS instances and connect URLs

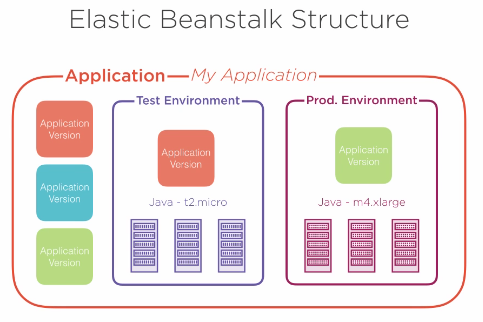
**Elastic Beanstalk** – app service that will take your app written in any language and get it running on an EC2 Instance.

**VPC** – service enabling you to secure instances in AWS. Allows you to ensure your data is safe.

**CloudFront** – CDN service that enables you to edge your files so uses can get to files fast. Cache.

**CloudWatch** – Alarm service to let you know when Instances are down, or when resources have issues.

**Elastic Beanstalk** – Adds a lot of convenience. Can be deployed from the web console, AWS CMD line or SDK. For environments, you can have a set it and forget it type of configuration. Also enables you to have aggregated monitoring and logging across multiple EC2 instances. Configuring each environment in Elastic Beanstalk is where I will spend most of my time, since you can set up the deployment, load balancing and scaling rules there.

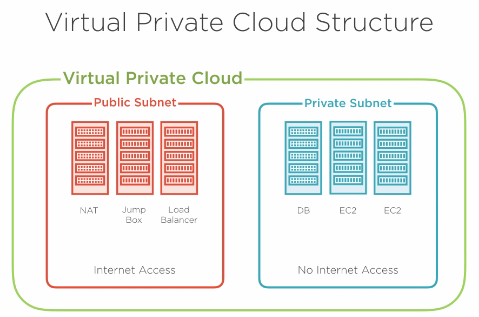


Each Application has a limit to 500 different versions. Monitoring is a great tool as well. Data is aggregated from all the instances. Metrics such as CPU, number of requests and network traffic. You can also pull the logs from all the instances to see if something is not working correctly.

**VPC – Virtual Private Cloud** – is used when launching EC2 instances, to secure and control access to them. VPC is the evolution of Security Groups. VPC lets you:

* Configure VPC Routing Tables
* Use NAT Gateways for Outbound Traffic
* Internal IP Address Allocation

Inside each Virtual Private Cloud is a Subnet. Subnets allow us to further group resources and assign different rules for each. One reason would be to have private and public subnets.



Subnets are another resource to increase security.

Two ways that VPC controls routing is with Routing Tables and Network ACL. Route tables allow you to control what goes where. You can override certain IP ranges and move them elsewhere.

Network Access Control Lists act as subnet level firewalls. Allowing or disallowing IP ranges for both incoming and outgoing connections. Basic VPC Configuration is free and therefore used.

**CloudWatch** – Solution for your monitoring needs.

Monitoring Resources and Acting on Alerts – 2 main functionalities

You can set your metrics and alarms for each item. You choose from a pre-existing set of metrics from each service. You then set a threshold alarm and a set of actions to take. Actions can be notifying you by email or SMS to triggering an Autoscale EC2 Instance. CloudWatch can also consume and monitor logs. Awslogs agent can go into your EC2 and get the log needed. It then sends the logs to CloudWatch.

* An example would be to track how many times a specific exception happens in the logs and then sending a notification once it happens a set amount of times.

**CloudFront** – Content delivery network that allows you to serve files globally with very fast connections.

CloudFront works with Route53, Load Balancer, EC2, and S3. You start by creating a distribution. A distribution is a set of content to be served by CloudFront. Once a distribution has been created a URL is given. Example: <https://d3nwl6hiok169.cloudfront.net> . You can configure http settings, edge locations or even SSL certificates.

**Web Console** – Spinning up new resources, configuring and monitoring those resources

**Software Development Kits (SDK)** – main way to program apps to interact with services in AWS

**AWS Command Line tool** – can do the same as web console but with the terminal

Web Console is the usual first starting point. AWS SDKs are code libraries to interact with AWS services.

Will I need to look into AWS SDKs more? \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AWS Command Line Interface or CLI

* Being able to use shell scripts are huge

aws <service> <command> <arguments>

docs.aws.amazon.com/cli/latest/reference/

For the command line, there is an installer with windows. If you have mac or linux, use python pip. There will be an initial setup configuration with access keys that I will need to set up.

aws dynamodb put-item –table-name aws-developer – item

SHELL SCRIPTS\*\*\*\*\*\*   
Command line is great for doing things quickly or automating.

Web Console – First Screen post login – 55 Services are categorized. Each service has its own dashboard, but they are relatively similar.

Command Line – Great or Shell Scripting, Interactive with Any Service. AWS CLI is made for Operations Engineers more than Developers.

We will use the Command Line to first Configure the local development environment.

AWS Access Key gives access for SDK and CLI

For a web app of pizza social media.

* EC2 for the app
* Images and static assets are in S3
* Users will be stored in RDS
* Once logged in, save the session in the cache with elasticache
* Toppings and modifications are saved to DynamoDB
* When Pizza is created it will be saved in DynamoDB
* The Picture of the new pizza is saved as an asset in S3

**Install the AWS CLI with pip in the command line. –**  pip install awscli

To check the version you are using use “aws –version”

Restarting the computer is necessary when using Windows 10 to ensure the path is working.

Console.aws.amazon.com create your account and login.

Generating an Access Key – required for both the CLI and SDK.

* Click on the user name in the top right corner and click on security credentials
* Access Keys and create new access key button. The info is then populated.

Back in the command line, enter aws configure. It then asks for your access Key ID and the Secret access key. Default Region name is the prefix for your region in the top right.  
\*\*\*\*\*What do I put for the default output format? JSON?

Once this is configured, I can type aws ec2 describe-instances

And I should see “Reservations”: [] letting me know theres nothing to start and its ready.

Continuous Integration – is a software development practice in which developers regularly merge their code changes into a central repository, after which automated builds and tests are run. This is a split mind-state between a cultural component, which basically means learning to integrate frequently) and the other entails both an automation component, a CI or build service.

* The key goals of continuous integration is to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.
* A challenge is to maintain a single source code repository, automating builds and automatic testing.