Cloud Formation Templates:

--------------------------------------

* **AWSTemplateFormatVersion:** Specifies the AWS CloudFormation template version.
* **Description:** A text string that describes the template.
* **Mappings:** A mapping of keys and associated values that you can use to specify conditional parameter values. This is CloudFormation’s version of a “case” statement.
* **Outputs:** Describes the values that are returned whenever you view your stack’s properties. This gets displayed in the AWS CloudFormation Console.
* **Parameters:**Specifies values that you can pass in to your template at runtime.
* **Resources:**Specifies the stack resources and their properties, like our EC2 instance. This is the *only* *required* property.

dry-run = noop mode. In terraform world, this is known as a terraform plan. In response, the AWS team has released something called [**Change Sets**](https://aws.amazon.com/blogs/aws/new-change-sets-for-aws-cloudformation/)**.**

$ aws cloudformation create-stack --stack-name example --template-body file://templates/single-instance.yml --parameters file://parameters/single-instance.json

$ aws cloudformation create-change-set --stack-name example --template-body file://templates/instance-and-route53.yml --parameters file://parameters/instance-and-route53.json --change-set-name changeset-

**Create-stack update-stack delete-stack describe-stack list-stacks**

**Kubernetes**

=========

1. Install Kubectl
2. Install Minikube

Minikube start

Kubectl run hello-minikube –image=gc.io/google/ehoserver:1.4 –port=8000

Kubectl expose deployment hello-minikube –type=NodePort

Kubectl get pod

Kubectl delete deployment hello-minikube

Minikube stop



Kubectl apply –f ./deployment.yaml

Kubectl expose deployment tomcat-deployment --type=NodePort

Minikube service tomcat-deployment --url

Kubectl get pods

Kubectl describe pod -- List all pods in all namespaces

Kubectl port-forward tomcat 500:600 🡪 forwards one or more local ports to a pod

Kubectl attach pod name container -- > attaches the process that is already running inside an existing container

Kubectl exec --it tomcat bash 🡪 execute a command on a container

Kubectl label pods tomcat healthy=false 🡪Updates the label over a resources

Kubectl run image --port=5701 🡪Run a particular image over a cluster

**Masters**

======

1. API

2. Schedular

3. ControllerManager

**Node**

=====

Node – Worker node

Pod

Kubelet – managing processing of the individual pod.

Kubeproxy – make sure that network services are exposed in each pod

Replicaset -- setting replica for deployment

Kubectl scale --replicas=4 deployment/tomcat-deployment

Kubectl get deployments -- list of deployments

Kubectl rollout status deployment tomcat-deployment– view status of deployments roll outs

Kubectl set image – image of deployment

Kubectl rollout history -- history of a roll out,including previous versions

--revision == 4

Labels – key/values attach to objects like pods

Selectors – way of expressing how to select objects based on their lables

1.equality-based

2.set-based

Node selector -- uses labels and selectors to choose which odes

Kubectl apply –f ./dep.yml

**Health checks:**

Readiness probe Liveness probe

**K8s**

===

1.gui

2.scalable

3.auto scaling is possible

4 cluster creation in complex

5.manual load balancing is needed

6.rolling updates progressively update the pods one after another ensure ha

7.automatic rollback case of failure

8.share volume b/n containers with in same pod

9. in built log and monitoring tool

**Swarm**

1.No gUI

2.auto scaling is possible with manual intervention

3.scalable,5times better scaling

4,cluster creation is easy

5.automatic load balancing is done

6.rolling updates progressively update the pods one after another ensure

7.no automatic rollback case of failure

8.share volume b/n containers in the node

9.third party log and monitoring tool

**Kubeadm.conf**

1.bring up the cluster

Kubeadmin init –pod network

Kubectl apply(join master with nodes) to enale the calico network

Kubectl get pods -- wide --all namespaces -🡪 to get all deployed namespaces

Kubectl proxy

Kubeadm reset 🡪 remove the assigned details to the node

Namespaces -- virtual clusters

**Views**

------

Admin

Work loads

Services and discovery view

Storage and config view

Secrets

**Architecture**

---------------

**Master**

======

Managing the k8’s cluster. Entry point for admin tasks

**1. API server**

Admin tasks with in master node. Validates and process requests

**2. Schedular**

Schedules the work in to different each worker node.

**3. Controller manager**

Manages non terminating controlling loops which regulates the state of the k8s state.Current state is same as the desired state.

**4.Etcd**

Distributive key-value store to store the cluster state.We can configure with master or configure externally

**Node**

=====

Controlled by the master node

**1.pod**

Scheduling unit in k8s.logical collection of one or more container

**2.kubeproxy**

Network proxy runs on each pod

**3.kubelet**

Runs on each containers, and make sure all the containers are in healthy state.Connects to the runtime

**4.container runtime**

Containers

**Service** 🡪 type of k8s resource is configured to forward requests to a set of pod. Have an ip address and this ip address routes to a healthy pod.

1.cluster ip -- default service,reachable with in cluster ip

2.node port 🡪 each node ip at static port,outside the clusters

3. load balancers 🡪 exposes the services externally it will automatically created

4.external name : map services to the contents of the external name field by returning a cname record with its value , No proxy of any kind is set up

**Replication Controller**

------------------------------

Managing the pod life cycle.

Specified no of pod replicas are running at any given point of time

Supports EQUAITY based selectors

EQUAITY based selectors : it looks only exact names/words

**Replica Sets**

----------------

How many replicas of pod should be running.replacemet of RC

Supports SET based selectors

It looks the sets

**Deployment Controllers**

===================

Upgrading higher version of RC.

1.Create Deployement

2.Update Dep

3.Rollback dep

4.Scale Depl

5.Pause Dep

Kubectl get rs 🡪 to get replica sets

Kubectl describe deployment

Kubectl rollout status deployment

Kubectl rollout history deployment

Kubectl rollout undo deployement

Kubectl scale deolyment

Kubecl autoscale deployment –min=3 cpu=

Kubectl get hpa(horizontal pods)

Kubectl delete hpa nginx-deployment

Kubectl rollout pause

Resume

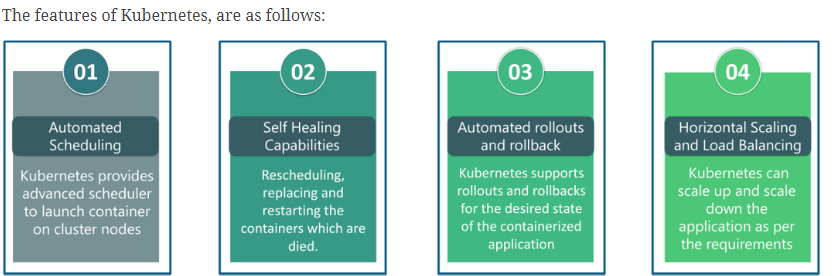
Kubectl set image nginx

Kubectl set resources depl

Kubernetes is an open-source container management tool which holds the responsibilities of container deployment, scaling & descaling of containers & load balancing.

It’s a known fact that Docker provides the lifecycle management of containers and a Docker image builds the runtime containers. But, since these individual containers have to communicate, Kubernetes is used.  So, Docker builds the containers and these containers communicate with each other via Kubernetes. So, containers running on multiple hosts can be manually linked and orchestrated using Kubernetes.

container orchestration means all the services in individual containers working together to fulfill the needs of a single server.



The fundamental behind Kubernetes is that we can enforce the desired state management, by which I mean that we can feed the cluster services of a specific configuration, and it will be up to the cluster services to go out and run that configuration in the infrastructure

Heapster is a cluster-wide aggregator of data provided by Kubelet running on each node.This container management tool is supported natively on Kubernetes cluster and runs as a pod, just like any other pod in the cluster.

Minikube is a tool that makes it easy to run Kubernetes locally. This runs a single-node Kubernetes cluster inside a virtual machine.

Kubectl is the platform using which you can pass commands to the cluster.  provides the CLI to run commands against the Kubernetes cluster with various ways to create and manage the Kubernetes component.

Kubelet is an agent service which runs on each node and enables the slave to communicate with the master. So, Kubelet works on the description of containers provided to it in the PodSpec and makes sure that the containers described in the PodSpec are healthy and running.

Node 🡪cluster worker machine.it provides necessary services to run the pods.It managed by the master

The Kubernetes Architecture has mainly 2 components – the master node and the worker node.

The master node has the kube-controller-manager, kube-apiserver, kube-scheduler, etcd. Whereas the worker node has kubelet and kube-proxy running on each node.

Kube-proxy can run on each and every node and can do simple TCP/UDP packet forwarding across backend network service.

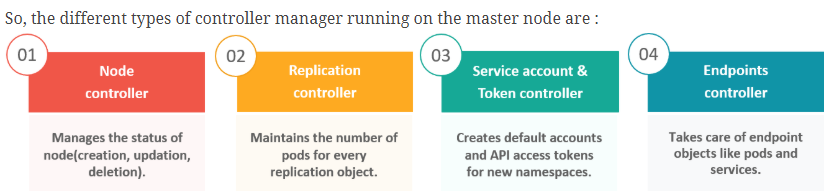
Kubernetes master controls the nodes and inside the nodes the containers are present

The kube – apiserver follows the scale-out architecture and, is the front-end of the master node control panel. This exposes all the APIs of the Kubernetes Master node components and is responsible for establishing communication between Kubernetes Node and the Kubernetes master components.

The kube-scheduler is responsible for distribution and management of workload on the worker nodes. So, it selects the most suitable node to run the unscheduled pod based on resource requirement and keeps a track of resource utilization. It makes sure that the workload is not scheduled on nodes which are already full.

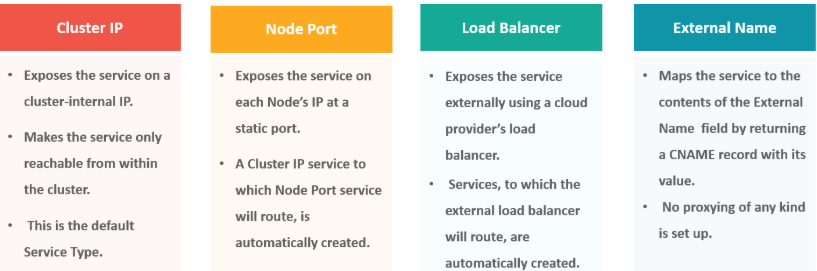
**Kubernetes controller manager?**

Multiple controller processes run on the master node but are compiled together to run as a single process which is the Kubernetes Controller Manager. So, Controller Manager is a daemon that embeds controllers and does namespace creation and garbage collection. It owns the responsibility and communicates with the API server to manage the end-points.



Etcd is written in Go programming language and is a distributed key-value store used for coordinating between distributed work. So, Etcd stores the configuration data of the Kubernetes cluster, representing the state of the cluster at any given point in time.

**types of services in Kubernetes?**



A load balancer is one of the most common and standard ways of exposing service. There are two types of load balancer used based on the working environment i.e. either the Internal Load Balancer or the External Load Balancer. The Internal Load Balancer automatically balances load and allocates the pods with the required configuration whereas the External Load Balancer directs the traffic from the external load to the backend pods.

**Kubernetes cluster data is stored in** Etcd

ReplicaSet and Deployment **is a Kubernetes Controller**

**The Kubernetes Network proxy runs on** All the nodes

 the responsibilities of a node controller?

1. To assign a CIDR block to the nodes
2. To maintain the list of nodes
3. To monitor the health of the nodes

the responsibilities of Replication Controller?

1. Update or delete multiple pods with a single command
2. Helps to achieve the desired state
3. Creates a new pod, if the existing pod crashes

**What did the 1.8 version of Kubernetes introduce?**

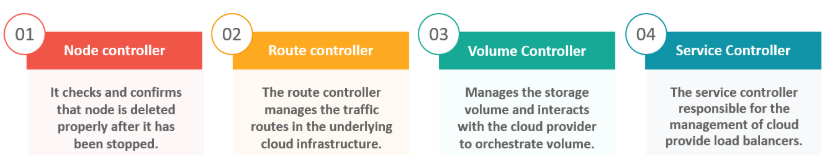
1. Taints and Tolerations

**The handler invoked by** **Kubelet to check if a container’s IP address is open or not is** TCPSocketAction

**How to define a service without a selector?** Specify the external name

Ingress network is a collection of rules that acts as an entry point to the Kubernetes cluster. This allows inbound connections, which can be configured to give services externally through reachable URLs, load balance traffic, or by offering name-based virtual hosting.

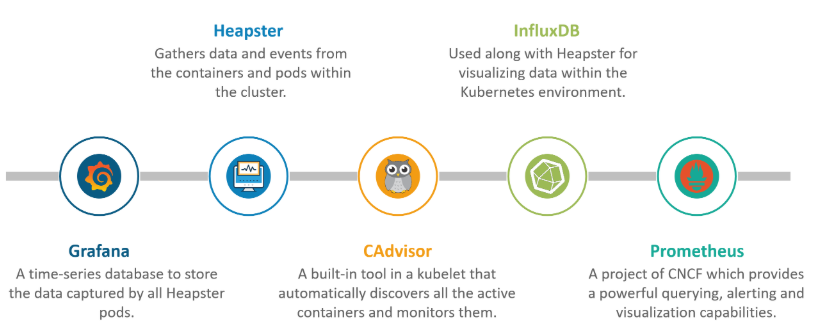
The Cloud Controller Manager is responsible for persistent storage, network routing, abstracting the cloud-specific code from the core Kubernetes specific code, and managing the communication with the underlying cloud services.  It might be split out into several different containers depending on which cloud platform you are running on and then it enables the cloud vendors and Kubernetes code to be developed without any inter-dependency. So, the cloud vendor develops their code and connects with the Kubernetes cloud-controller-manager while running the Kubernetes.



Container resource monitoring?

As for users, it is really important to understand the performance of the application and resource utilization at all the different abstraction layer, Kubernetes factored the management of the cluster by creating abstraction at different levels like container, pods, services and whole cluster. Now, each level can be monitored and this is nothing but Container resource monitoring.

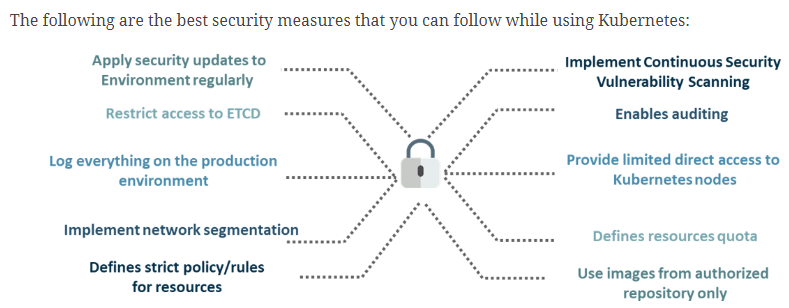
The various container resource monitoring tools are as follows:



Replica Set and Replication Controller do almost the same thing. Both of them ensure that a specified number of pod replicas are running at any given time.

Replica Set use Set-Based selectors while replication controllers use Equity-Based selectors.

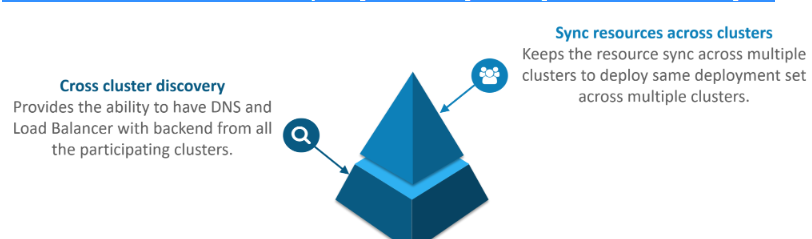
* Equity-Based Selectors: This type of selector allows filtering by label key and values. So, in layman terms, the equity-based selector will only look for the pods which will have the exact same phrase as that of the label.  
  Example: Suppose your label key says app=nginx, then, with this selector, you can only look for those pods with label app equal to nginx.
* Selector-Based Selectors: This type of selector allows filtering keys according to a set of values. So, in other words, the selector based selector will look for pods whose label has been mentioned in the set.  
  Example: Say your label key says app in (nginx, NPS, Apache). Then, with this selector, if your app is equal to any of nginx, NPS, or Apache, then the selector will take it as a true result.



Headless Service is similar to that of a ‘Normal’ services but does not have a Cluster IP. This service enables you to directly reach the pods without the need of accessing it through a proxy.

Multiple Kubernetes clusters can be managed as a single cluster with the help of federated clusters. So, you can create multiple Kubernetes clusters within a data center/cloud and use federation to control/manage them all at one place.

The federated clusters can achieve this by doing the following two things. Refer to the below diagram.



As the company’s goal is to shift from their monolithic application to microservices, they can end up building piece by piece, in parallel and just switch configurations in the background. Then they can put each of these built-in microservices on the Kubernetes platform. So, they can start by migrating their services once or twice and monitor them to make sure everything is running stable. Once they feel everything is going good, then they can migrate the rest of the application into their Kubernetes cluster.

AWS

====

* 1. **Compute :** 
     1. EC2

Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Resources:

------------------

Running instances

Dedicated hosts

Volumes

Key Pairs

Placement Groups

Elastic Ip’s

Snapshots

Load Balancers

Security Groups

**Launch Instance:**

* 1. ways
     1. Launch Instance
     2. Launch from template

**Launch Instance :**

* + - * 1. Instance State : Start,Stop,Reboot,Terminate
        2. Instance Settings : Add/Edit Tags, Attach to auto scaling group,Attach/Replace IAM Role,Change Instance Type,Change Termination Protection,view/change User data,Change ShutDown behavior,**Get System Log**,
        3. Create Image : AMI
        4. Networking : Change Security Groups,Manage IP address,attach network interface,detach NI,
        5. CloudWatch monitoring : Enabled/Disabled Cloud watch montoring,

**Status checks**

1. System status
2. Instance status

An Amazon EC2 Dedicated host is a physical server with EC2 instance capacity dedicated for your use and allows you to reliably launch EC2 instances on the same Dedicated host over time.

Dedicated Instances - You pay for the instances, but they get placed on whatever dedicated hardware Amazon decides.

Dedicated Host - You pay for the entire physical server and can, in effect, run instances on it as you please.

Events may include operational activities planned by AWS such as instance reboots or retirements

EC2 reports provide detailed usage or billing data for your EC2 instances. These reports provide you with insight into how you are using Amazon EC2

Instance Types

---------------------

On-demand

Reserved

Spot

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Running On-Demand EC2 instances |  | https://d1z9niso53l8pm.cloudfront.net/2018-12-03T10-41-48_38d8fd9830a2481d70d2196f25a967dc5b6c7ffd21aea9824518578c1f1f775b/Static/clear.cache.gif |   /spot 20 | 20 |

EBS Limits 300

Networking -🡪 Elastic IP’s 5

VPC -- 5

Subnetcs – 200

Network acls 200

SG per region --- 2500

Active vpc peering connections 50

Route tables per vpc 200

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Entries per route table |  | https://d1z9niso53l8pm.cloudfront.net/2018-12-03T10-41-48_38d8fd9830a2481d70d2196f25a967dc5b6c7ffd21aea9824518578c1f1f775b/Static/clear.cache.gif |   50 | 50 |

Auto Scaling Limits 200

Load Balancing Limits(20) tg 3000

**On-Demand instances** let you pay for compute capacity by the hour with no long-term commitments.

**Spot instances** enable you to bid on unused EC2 instances, which can lower your Amazon EC2 costs significantly.

**Reserved :** aws reservedinstances

You can create **flow logs** on your resources to capture IP traffic flow information for the network interfaces for your resources.

Elastic Load Balancer:

-----------------------------

* 1. types

1 . Classic LB

**PREVIOUS GENERATION**

for HTTP, HTTPS, and TCP

Choose a Classic Load Balancer when you have an existing application running in the EC2-Classic network.

2 . Application LB

http,https

Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including micro services and containers.

. Networknlb

Tcp

Choose a Network Load Balancer when you need ultra-high performance and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second while maintaining ultra-low latencies.

Step 1: Configure Load Balancer Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network.

Scheme : internet facing(default),internal

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.

Step 2: Configure Security Settings(https)

Step 3: Configure Security Groups

A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balance

Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify, and performs health checks on the targets using these health check settings. Note that each target group can be associated with only one load balancer.

Target group

**Type** : instance

Ip

Lambda function

Protocol : http/https

Port : 80

## Health checks

Protocol :

Path

Advanced Health Checking settings:

Port : traffic port,override

Health threshold : 5

Unhealthy threshold : 2

Timeout : 5 sec

Interval : 30 sec

Success code : 200

Step 5: Register Targets

Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

**Attributes:**

Access Logs : S3 bucket/Prefix

Integration Services

-------------------------

### AWS Config : ConfigRecords configuration changes to your load balancer and provides visibility into its integration with other resources

### AWS WAF : Protects your web applications by enabling AWS WAF directly on your load balancer.

Auto scaling

---------------

Auto Scaling to manage Amazon EC2 capacity automatically, maintain the right number of instances for your application, operate a healthy group of instances, and scale it according to your needs

Step 1: Create or select a launch template/Configuration

Create or select the launch template that your Auto Scaling group will use to launch your EC2 instances.

Step 2: Create Auto Scaling group

Next, give your group a name and specify how many instances you want to run in it.

* Choose AMI
* Choose Instance Type
* Create Launch Configuration
* Add Storage
* SG’s

Step 2: Create Auto Scaling group

* + 1. ASG group details
    2. Configure Scaling Policies

Metric type : avg cpu utilization,network in,network out

Increase group size/ Decrease Group size

Execute policy when New Alarm

Take the action

Step 3 : Configure Notifications

Step 4 : Configure tags

**S3**

**----**

**Prop’s:**

**1.Versioning :** keep multiple versions of an object in the same bucket

Enabled and suspended

2.Server Access Logging : set up access log records that provide details about access requests

3. Static Web site hosting

4.Object level logging

5. Default encryption

6. Object lock: prevent objects from being deleted

7. Tags: to track your cost against projects

8. Events: receive notification when specific events occur in u r bucket

9. Requester Pays: requester will pay requests and dta transfer

10. Transfer acceleration: enable fast, easy and secure transfers of files to and from your bucket

**Permissions :**

--------------------

1 . **Public access settings**

a)Manage public access control lists

b) Manage public bucket policies

2 . **Access Control List**

Access for u r ac

Access for other aws ac’s

Public Access

S3 log delivery group

* 1. Bucket policies
  2. CORS configuration(Cross origin resource sharing)

Storage Life Cycle

------------------------

Standard Frequently access data

Intelligent-Tiering Long lived InFrequently access data

Standard-IA Long lived InFrequently access

One Zone –IA Long lived InFrequently access

Glacier data archiving from retrieval times ranging from minutes to hours

Reduced Redundancy (not recommended) Frequently access data non critical data

Create Bucket

------------------

1.Region and Name

2.Configure Options :

Versioning, server access logging(Log requests for access to ur bucket)

Tags, Object-Level Logging,

**CloudWatch request metrics**

Management:

-------------------

Life cycle

Replication

Analytics

Metrics

Inventory

Inventory

You can receive a file of your object inventory on a daily or weekly basis for the entire bucket or a shared prefix.

**EFS**

-----

Amazon EFS provides file storage for use with your EC2 instances.

## 1 . Configure file system access

Create mount targets

## 2. Configure optional settings

Add tags

**Choose performance mode:**

We recommend **General Purpose** performance mode for most file systems. **Max I/O** performance mode is optimized for applications where tens, hundreds, or thousands of EC2 instances are accessing the file system.

**Choose through put mode**

We recommend **Bursting** throughput mode for most file systems. Use **Provisioned** throughput mode for applications that require more throughput than allowed by **Bursting** throughput

**Glacier:**

-----------

Amazon S3 Glacier is an extremely low-cost storage service that provides secure, durable, and flexible storage for data backup and archival.

Data is stored in S3 Glacier in "archives." A single archive can be as large as 40 terabytes.

Vaults allow you to organize your archives and set access policies and notification policies. Get started by giving your vault a name

**Storage Gateway**

----------------------

The AWS Storage Gateway is a service connecting an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and AWS's storage in frastructure.

**RDS**

--------

1. Easily set up, operate, and scale a relational database in the cloud.

VPC : A virtual private cloud (**VPC**) is a virtual network dedicated to your **AWS** account.

Internet Gateway : Creates an **internet gateway** for use with a VPC. After creating the **internet gateway**, you attach it to a VPC using AttachInternetGateway

Route Table : A **route table** contains a set of rules, called **routes**, that are used to determine where network traffic is directed. Each subnet in your VPC must be associated with a **route table**; the **table** controls the**routing** for the subnet.

**NAT Gateways**. You can use a network address translation (**NAT**) **gateway** to enable instances in a private subnet to connect to the internet or other **AWS** services, but prevent the internet from initiating a connection with those instances. ... You are charged for creating and using a **NAT gateway** in your account

**Network ACLs**. A **network** access control list (**ACL**) is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. You might set up **network ACLs** with rules similar to your security groups in order to add an additional layer of security to your VPC. IN Bound / Out Bound Rules/subnet Associations

Route53

----------

You can use Amazon Route 53 to register new domains, transfer existing domains, route traffic for your domains to your AWS and external resources, and monitor the health of your resources.

DNS Management

Traffic Management

Domain Registration

Availability Monitoring

**SNS:**

-----

Amazon Simple Notification Service (SNS) is a fast, flexible, fully managed push messaging service.  
Create topic

Create a communication channel to send messages and subscribe to notifications

Create subscription

Subscribe an endpoint to a topic to receive messages published to that topic

[Publish message](https://us-east-2.console.aws.amazon.com/sns/v2/home?region=us-east-2" \l "/publish)

[Publish a message to a topic or as a direct publish to a platform endpoint](https://us-east-2.console.aws.amazon.com/sns/v2/home?region=us-east-2" \l "/publish)

Code Commit

-----------------

**AWS CodeCommit** is a secure, highly scalable, managed source control service that hosts private Git repositories. **AWS CodeCommit** eliminates the need for you to manage your own source control system or worry about scaling its infrastructure.

Code Build :

**AWS CodeBuild**, a fully managed build service that compiles source code, runs tests, and produces software packages that are ready to deploy.

**Code Pipeline:**

**AWS CodePipeline** is a continuous delivery service that enables you to model, visualize, and automate the steps required to release your software. Walks through how to set up **AWS CodePipeline** and integrate it with other services.

**AWS CodeDeploy** is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon **EC2**,**AWS** Fargate, **AWS** Lambda, and your on-premises servers. ... You can use **AWS CodeDeploy** to automate software deployments, eliminating the need for error-prone manual operations.

**Monitoring (Basic/Detailed)**

1. Status Checks
2. Networks(In/Out/Packets In/Out)
3. CPU Utilization
4. Disk(Read/Write)
5. Volume(throughput/read/write/latency)gp2/provisioned iops/magentic

A tag consists of a case-sensitive key-value pair.

**Service Health:**

Service Status

AZ Status

Volume:

========

Id,size,Type,Snapshot

* 1. Modify Volume : size,type
  2. Create snapshot
  3. Attach/detach/Force detach/Delete Volume

Can I attach more than 2 volumes in to same instance?

No, according to the EBS docs: "A volume can only be attached to one instance at a time".

Create Volume : Using Snapshot,Create new volume

Using volume can able to create snapshot(Volume mandatory).

Using snapshot can able to create volume and Image

Block Device Mappings:

--------------------------------

Complete Block Device

Root Device Mappings

KeyPair

--------

Having Private key in our machine

* + 1. ECS
    2. EKS
    3. Lambda
    4. Elastic Bean Stalk
    5. ECR

**AWS Secrets Manager** helps you protect access to your applications, services, and IT resources. You can easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.

**Amazon GuardDuty**

Intelligent threat detection to protect your AWS accounts and workloads.

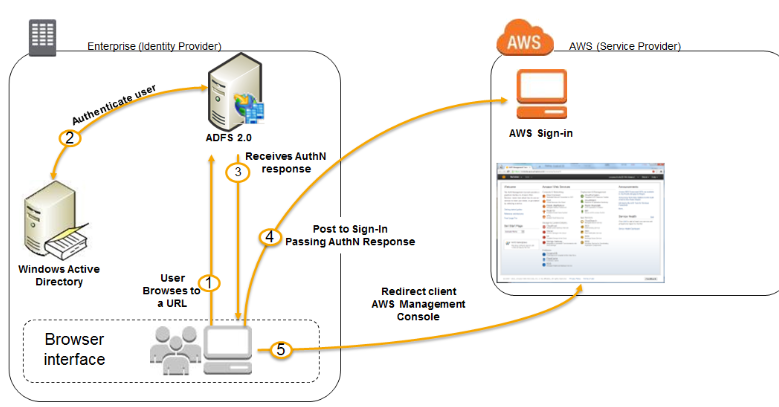
When you enable GuardDuty, you grant GuardDuty permissions to analyze AWS CloudTrail logs, VPC Flow Logs, and DNS query logs to generate security findings. [Amazon GuardDuty](https://aws.amazon.com/guardduty/) is a managed threat detection service that continuously monitors for malicious or unauthorized behavior to help you protect your AWS accounts and workloads.

**AWS Organizations**

AWS Organizations enables you to centrally apply policy-based controls across multiple accounts in the AWS Cloud.You can consolidate all your AWS accounts into an organization, and arrange all AWS accounts into distinct organizational units.

SAML(Security Assertion Markup Language)-compliant IdP(identity provider).

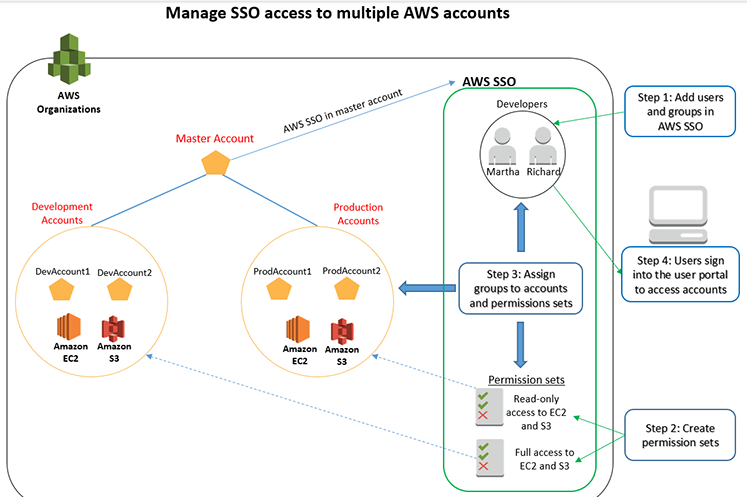
AD FS (Active Directory Federation Services).



Amazon Web Services (AWS) supports only IdP(Identity Provider) initiated Single Sign On(SSO)

AWS Single Sign-On is a cloud service that makes it easy to manage SSO access to multiple AWS accounts and business applications.

 Example.com has five AWS accounts: a master account (called MasterAcct), two developer accounts (DevAccount1 and DevAccount2), and two production accounts (ProdAccount1 and ProdAccount2). Example.com uses [AWS Organizations](https://aws.amazon.com/organizations/) to manage these accounts and has already [enabled AWS SSO](https://docs.aws.amazon.com/singlesignon/latest/userguide/step1.html).



AWS Certificate Manager (ACM) makes it easy to provision, manage, deploy, and renew SSL/TLS certificates on the AWS platform

* Communication between the browser and the website are on SSL (Secure Sockets Layer) Port 443

**Amazon Cognito** offers user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools provide AWS credentials to grant your users access to other AWS services.

**Amazon Simple Queue Service (SQS)** is a reliable, scalable, fully-managed message queuing service.

Standard(sns topics & lambda)

Fifo(lambda doesn’t support)

https://blog.sourcerer.io/full-guide-to-developing-rest-apis-with-aws-api-gateway-and-aws-lambda-d254729d6992

An API is a URL that you can perform GET, PUT, POST, and DELETE requests on to control another service. Common uses are providing database control, performing actions on third-party APIs (API-ception), or controlling another service.

Providing an API also allows you to restrict who is able to access the database or run the code. You can do this by requiring an API key. This key is used to identify the user making the request, and then allowing or rejecting the request.

The first thing we need to do is to add a resource onto the API. Using resources allows us to group similar API calls together using nested slashes.

Now that we have length and genre, we can create methods for getting and adding data to a table.

We are able to respond to these API requests using Lambdas, which is great as they only run when we need them to.

**AWS Lambda**

* Environment Variables, obviously meant for configuration settings
* Tags, meant for organizing your Functions
* Execution Roles, meant for permissions
* Basic Settings, including maximum memory and execution time
* Network, for network parameters
* Debugging
* Concurrency, meaning the maximum number of instances that will spawn

 In the subsequent window click Create new test event and give an Event Name.

A Lambda function can be triggered by any of a long list of event sources

using the AWS API Gateway, to invoke a simple Lambda function that handles GET, POST and DELETE requests.

REST services take actions corresponding to the HTTP request verb. These four are meant to correspond to the CRUD model, with POST equating to Create, GET equating to Read, PUT equating to Update, and DELETE equating to Destroy.

This Lambda function violates the principle that Lambda functions must be stateless. Since Lambda functions are automatically scaled to handle the traffic, there could be hundreds or thousands of myRESTservice containers

Do not create the OPTIONS Method, instead it is auto-created when we enable CORS support. For the /service selection, use the Actions dropdown to create a Resource. Then for DELETE, GET and POST, use the dropdown to create Methods

“Resource” is the word used in API Gateway to describe a given URL path, while “Method” is used in API Gateway to describe the HTTP verbs on a given Resource.

 Conceptually, each position in the URL structure is a Resource that is managed by HTTP-based Methods.

The next step is to configure each of the methods to connect with the Lambda function. Let’s start with the POST verb.The next step is to click on Body Mapping Templates and to configure how the POST request body is handled. Click on Add Mapping Template to add an application/json choice

select the *Method Request Passthrough* template.This template automatically passes through a bunch of useful data, and sets up the *event*object to match the code we wrote previously.

**Validating data in the POST method**

As we said earlier, the POST operation should check its input for validity. For example our Resource URL is /service, so POST should be given a Service object.

One advantage of the API Gateway service is it uses OpenAPI (a.k.a. Swagger) API definitions. These include online tools like the Swagger Editor, as well as code generation tools. From an OpenAPI spec we can not only deploy an API to the API Gateway, but we can generate a standalone implementation to run on our own server.

However, throughout the API Gateway UI it talks about importing and exporting Swagger v2 specifications. The OpenAPI Initiative released OpenAPI v3 in late 2017, so clearly API Gateway has yet to be updated to match.

**Amazon ECS** makes it easy to deploy, manage, and scale Docker containers running applications, services, and batch processes. Amazon ECS places containers across your cluster based on your resource needs and is integrated with familiar features like Elastic Load Balancing, EC2 security groups, EBS volumes and IAM roles.

[**ECS**](https://aws.amazon.com/ecs/) is the AWS Docker container service that handles the orchestration and provisioning of Docker containers.

Summary of the ECS Terms:

First we need to cover ECS terminology:

**Task Definition** — This a blueprint that describes how a docker container should launch. If you are already familiar with AWS, it is like a LaunchConfig except instead it is for a docker container instead of a instance. It contains settings like exposed port, docker image, cpu shares, memory requirement, command to run and environmental variables.

**Task —**This is a running container with the settings defined in the Task Definition. It can be thought of as an “instance” of a Task Definition.

**Service —**Defines long running tasks of the same Task Definition. This can be 1 running container or multiple running containers all using the same Task Definition.

**Cluster —**A logic group of EC2 instances. When an instance launches the ecs-agent software on the server registers the instance to an ECS Cluster. This is easily configurable by setting the ECS\_CLUSTER variable in /etc/ecs/ecs.config described [here](http://docs.aws.amazon.com/AmazonECS/latest/developerguide/launch_container_instance.html).

**Container Instance** — This is just an EC2 instance that is part of an ECS Cluster and has docker and the [ecs-agent](https://github.com/aws/amazon-ecs-agent" \t "_blank) running on it.

Container definition

* Choose an image for your container below to get started quickly or define the container image to use.

## Task definition

* A task definition is a blueprint for your application, and describes one or more containers through attributes. Some attributes are configured at the task level but the majority of attributes are configured per container.

## Define your service

* A service allows you to run and maintain a specified number (the "desired count") of simultaneous instances of a task definition in an ECS cluster.

Configure your cluster

* The infrastructure in a Fargate cluster is fully managed by AWS. Your containers run without you managing and configuring individual Amazon EC2 instances.

Amazon Elastic Container Registry (ECR) is a fully-managed container registry that makes it easy for developers to store, manage, and deploy container images.

**Kubernetes related:**

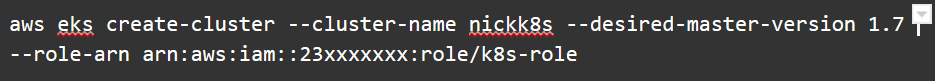
* kubectl: the command line tool for interacting with the Kubernetes API
* RBAC: Role Based Access Control. Similar to IAM, but controls access to Kubernetes resources (e.g. creating pods) rather than to AWS resources (e.g. writing to S3).
* Pod: the standard unit of Kubernetes deployments — can contain one or more containers. Also a group of whales.

#### IAM Roles for Team Members to use a Kubernetes Cluster

* There is an aws-auth-cm.yaml manifest file that the getting started tutorial tells you how to make. It turns out if things are specified properly in this file, EKS will be able to map your IAM roles to Kubernetes groups which have permissions. In our first attempt, we made a cortico:devs RBAC group and assigned our dev team’s IAM profiles to it.

Kubernetes API is extremely powerful. This API can be thought of as like a single abstraction layer that can help abstract resources both within AWS and on-premise.

If a user wants to create a cluster then he/she can use the below command line:



If a user wants to describe cluster details and get the metadata, he/she can use the below command line.

AWS EKS Demo 

To list the clusters, use the below command line.

AWS EKS Demo 

To delete the clusters, use the below command line.  
AWS EKS Demo 

EKS provides all API logs to CloudTrail. When a user calls eks create-cluster, this will go into CloudTrail. Just like any other AWS service, all the logs based on API layer will be in CloudTrail.

Things like kube-api server logs, the kube-scheduler logs, the kube-controller manager logs, etc. will be available in CloudWatch logs so that a user can aggregate the logs (like ELK stack) and monitor them