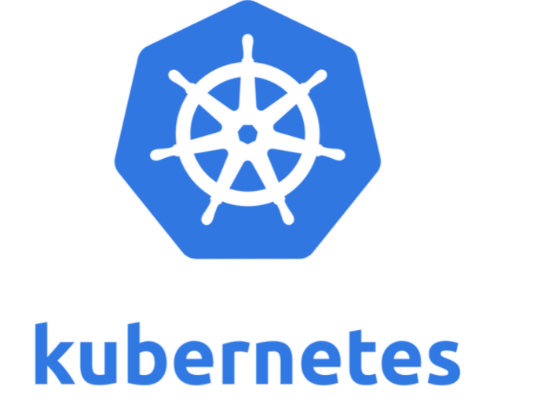
**REPORT SUBMITTED ON**

1. Kubernetes
2. SAML, OAuth, OpenId
3. IdP and SP

|  |  |  |
| --- | --- | --- |
| **S. No** | **Topic** | **Page No.** |
|  | Kubernetes | 2 |
|  | Pods and Nodes | 2-3 |
|  | Features of Kubernetes | 3 |
|  | SAML | 4 |
|  | IdP and SP | 4 |
|  | How IdP and SP works in SAML? | 5 |
|  | Advantages of SAML | 6 |
|  | OAuth | 6 |
|  | Principles of OAuth 2.0 | 6 |
|  | OAuth2.0 Roles | 7 |
|  | OAuth 2.0 Access Tokens and Authorization Code | 7 |
|  | OpenId | 7 |

**Submitted By**: Chirag Gaur

**Kubernetes:**

****

* Kubernetes is a container orchestration tool/engine which manages containerized application. It was developed by Google Lab and later donated to CNCF. It is open source and written on Golang
* Container orchestration tool/engine automates deploying, scaling and managing containerized application on a group of servers. It manages application which are available on container platform like Docker.
* Container allows a developer to package up an application with all the parts that it needs such as dependencies and libraries and ships it all as one package using a container platform like Docker. Docker is tool designed to deploy and run application using container.
* Organizations have to use multiple containers in order to ensure availability, load balancing and Scale up and down based on user load.

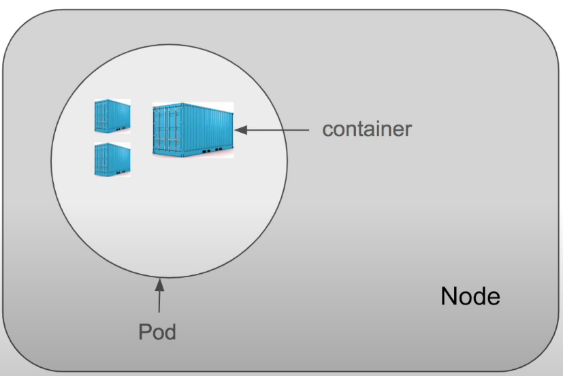
**Pods and Nodes:**

In Kubernetes we do not interact directly with the container. So, containers are wrapped in a higher-level structure or functional unit known as Pods. A Pod can have single container or multiple containers. Pods are housed into Node so which means a node can have single Pod or multiple Pods.

A pod contains the following

1. An application container
2. Storage resources
3. Unique IP

When we specify a pod, we can optionally specify how much CPU and RAM does the container needs. When container have resource requests specified, the scheduler can make better decisions about which nodes to place Pods on.



**Features of Kubernetes:**

1. *Automatic Bin Packing:* Kubernetes automatically packages your applications and schedules the container based on resources available like CPU and RAM. It packs all the pieces into few bins as possible in most efficient way.
2. *Service discovery and load balancing*: Kubernetes service is a set of pods working together. Kubernetes provides a single DNS name and its own IP address to set of pods. With this system Kubernetes have control over network and communication between pods and it can also distribute the load using load balancer.
3. *Storage Orchestration*: Container running inside a pod may need to store data. For this purpose, pods have storage volume and a single volume is shared with all the containers in the pod. Kubernetes allow to choose the type of storage such as local, cloud or network.
4. *Self-Healing*: Kubernetes have user defined health check. In case any container fails so it kills the container.
5. *Automated rollout and rollbacks*: Rollout means deploying the changes to the application or its configuration. Rollback means revert the changes and restore the application to previous state.

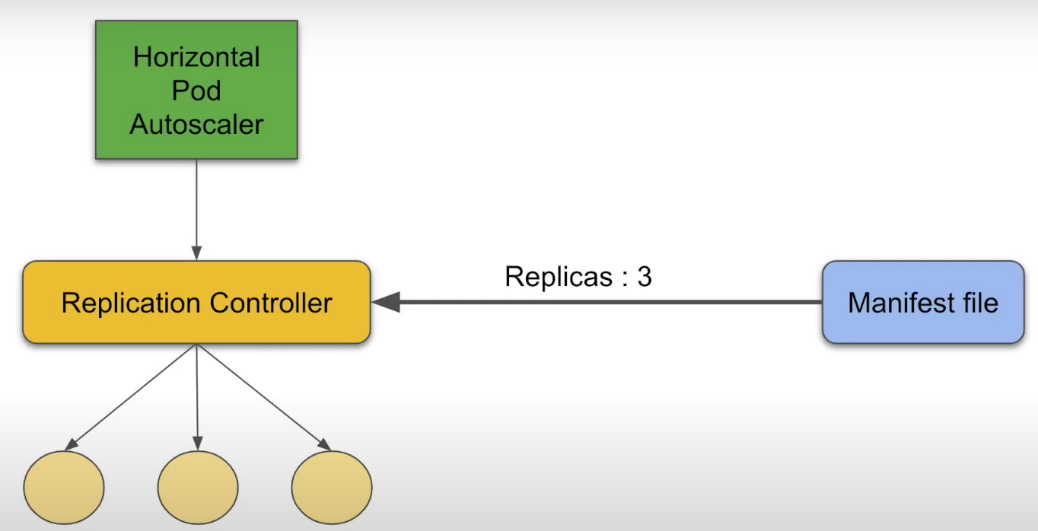
* Kubernetes progressively rollout the changes to application while monitoring application health to ensure it doesn’t kill all the application instance at the same time. If something goes wrong then changes will be roll back by the Kubernetes.

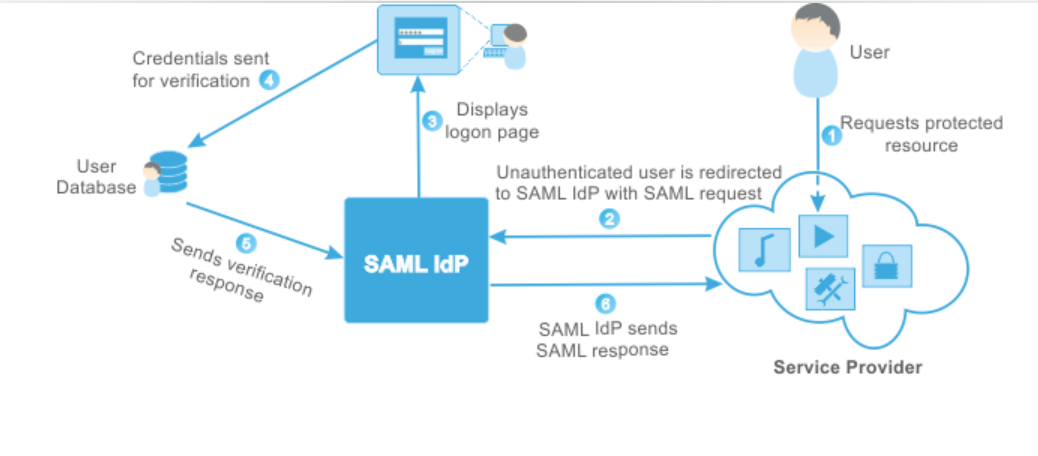
1. *Secret and Configuration Management:* In Kubernetes secret is helpful in handling sensitive data (like password or keys) and Conf Map is helpful in handling configuration. Secret is a Kubernetes object that separates sensitive data from pods. Conf Map is also a Kubernetes object that separates configurations from pods.

* Kubernetes manages secrets and configuration details for an application separately from the container image.
* Secrets and configuration of an application can be easily updated and deployed with rebuilding your image.

1. *Batch Execution*: Batch jobs require a process to be run to completion. In Kubernetes run to completion jobs are used for batch processing. Each job may require one or more pods. During job execution if container or pods fail then Job controller will reschedule the pod on another node. As the job is completed the pod will move from running to shut down state.
2. *Horizontal Scaling*: Scaling up or down of containers can be done in Kubernetes using commands or Kubernetes ui.

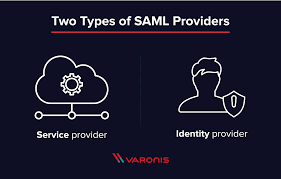
Replication controller is a structure that enables to create multiple pods. If a pod crashes then RC will replace it. Manifest file provides the information to Replication Controller about the no of pods to be run. Autoscaler scales the number of pods based on CPU utilization

****

**SAML:**

* SAML stands for Security Assertion Markup Language. It is a XML based standard which allows users to log into multiple applications using single sign on option.
* As the user’s identity is stored in directory so it would make more sense to fetch those identities from the active directory and log users into different applications.
* E.g: In Deloitte username and password remains stored in the directory and the target applications such as Skype, outlook, teams etc. will be logged in automatically using the same credentials for all of them.
* SAML transfers the user’s identity among the two parties. These two parties are identity provider and service provider.
* **Identity Provider (IdP)**- Identity provider is a centralized system where user credentials are stored. It is a system which authenticates user credentials. It authenticates the user's identity and authorizes to the service provider.
* **Service Provider (SP)**- Service provider provides services to end user. It authorizes the given user to access the requested resource. Service providers do not authenticate users but instead request authentication decisions from an identity provider.

**How IdP and SP works together in SAML?**

****

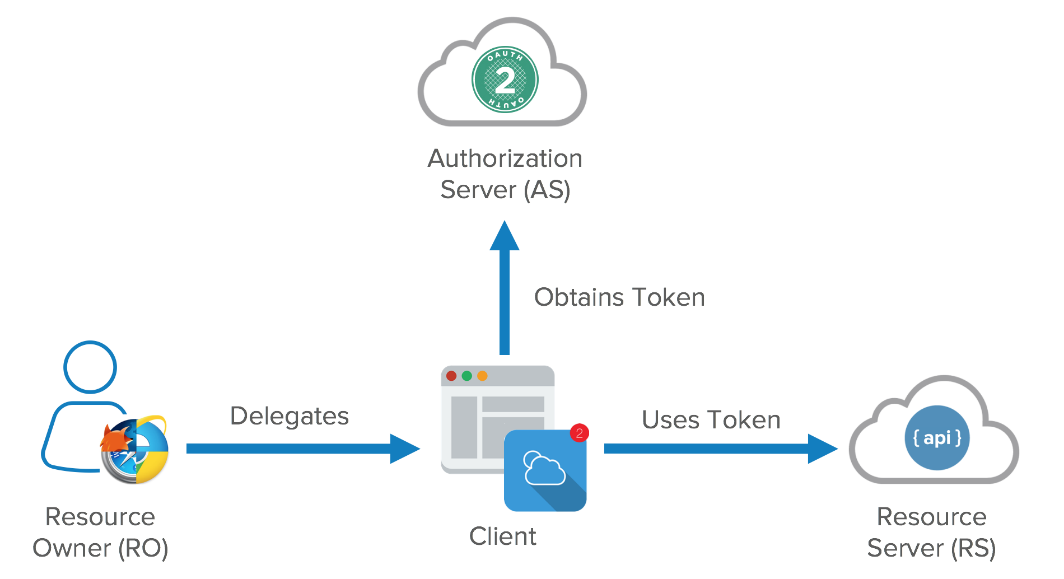
Let’s assume that a user is logged into a system that acts as an identity provider. The user wants to log in to a remote application, such as a support, finance or accounting application All these remote applications acts as Service providers. The following happens:

1. The user accesses the remote application using a link on an intranet, a bookmark, or similar and the application loads.
2. The application identifies the user’s origin (by application subdomain, user IP address, or similar) and redirects the user back to the identity provider, asking for authentication. This is the authentication request.
3. The user either has an existing active browser session with the identity provider or establishes one by logging into the identity provider.
4. The identity provider builds the authentication response in the form of an XML-document containing the user’s username or email address, signs it using an X.509 certificate, and posts this information to the service provider.
5. The service provider, which already knows the identity provider and has a certificate fingerprint, retrieves the authentication response and validates it using the certificate fingerprint.
6. The identity of the user is established and the user is provided with app access.

**Advantages of SAML:**

* Enhanced user experience
* No need to remember passwords
* Increased Security

**OAuth:**



* OAuth stands for Open Authorization. It is a standard protocol which allows an application or website to have access to resources which are hosted by some other application.
* It allows a client application to perform some actions on the resources on behalf on user without having user’s credentials.

**Principle of OAuth 2.0:**

* The principle of OAuth 2.0 is based on authorization. OAuth is designed for granting access to a set of resources.
* OAuth uses access tokens to perform authorization. Access token is a data which allows to perform authorization in order to gain access of resource on behalf of the end user.

**OAuth 2.0 Roles:**

OAuth roles define the essential components of an OAuth 2.0 system and are described as follows: -

1. *Resource Owner* is the one who owns the protected resources and grants access to them.
2. *Client* is the one who wants to get the access of some resources. The Client must hold the appropriate Access Token in order gain access to the resources.
3. *Authorization server* receives the request for access token from client and after successful authentication, it grants them the access token.
4. *Resource server* protects the user’s resources and receives access requests from the Client. It accepts and validates an Access Token from the Client and returns the appropriate resources to it.

**OAuth 2.0 Access Tokens and Authorization Code:**

The OAuth 2 Authorization server may not directly return an Access Token after the Resource Owner has authorized access. Instead, and for better security, an Authorization Code may be returned, which is then exchanged for an Access Token.

**OpenId:**

* It is an extra identity layer added on top of the OAuth. It allows to check the identity of the user based on authorization done by authorization server.
* In this we can fetch additional info about end user