UPES

Xebia

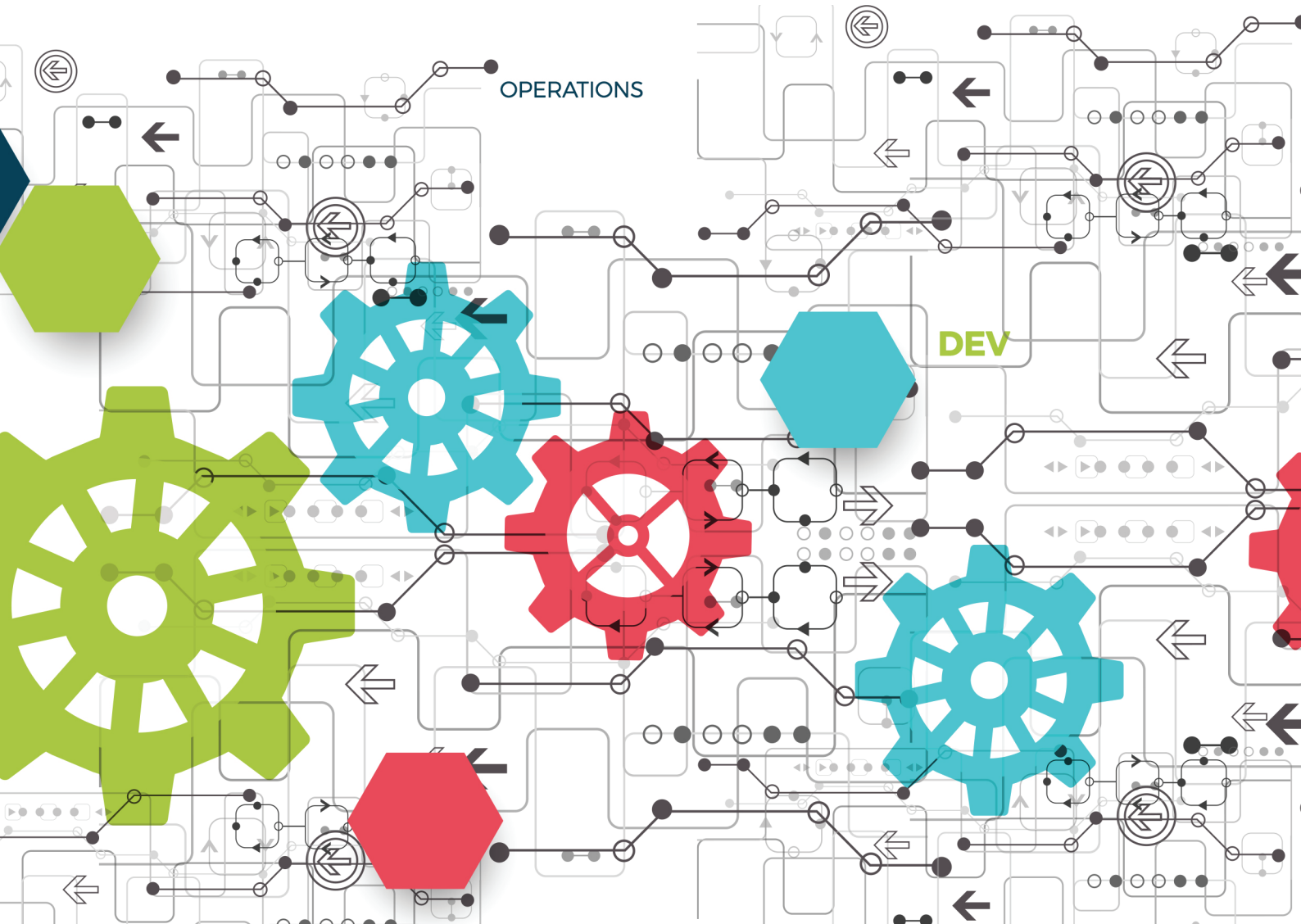
YEAR 3

B.Tech Computer Science and Engineering in DevOps

Application Containerization

MODULE 3

Orchestration Tools



Contents

Module Objectives **1**

Module Topics **2**

3.1 What is Orchestration? 2

3.2 Case study: Need of Orchestration 3

3.2.1 Need of Orchestration: Container and Microservices 5

3.3 Orchestration Tools 6

3.4 Docker Swarm 6

3.5 Docker Swarm Architecture 7

3.6 Kubernetes 8

3.6.1 Kubernetes Architecture 9

3.7 Amazon Web Services Elastic Container Services 10

3.8 AWS Elastic Container Services Architecture 10

3.9 AWS Elastic Kubernetes Services(EKS) 11

3.9.1 AWS Elastic Kubernetes Services(EKS) Architecture 12

3.10 Azure Kubernetes Services 12

3.11 Openshift 13

3.12 Google Kubernetes Engine 14

3.13 Kubernetes on cloud 14

3.14 Why we Need to Monitor Containers 15

3.15 What Needs to be Monitored in Containers? 16

3.15.1 Log Monitoring 16

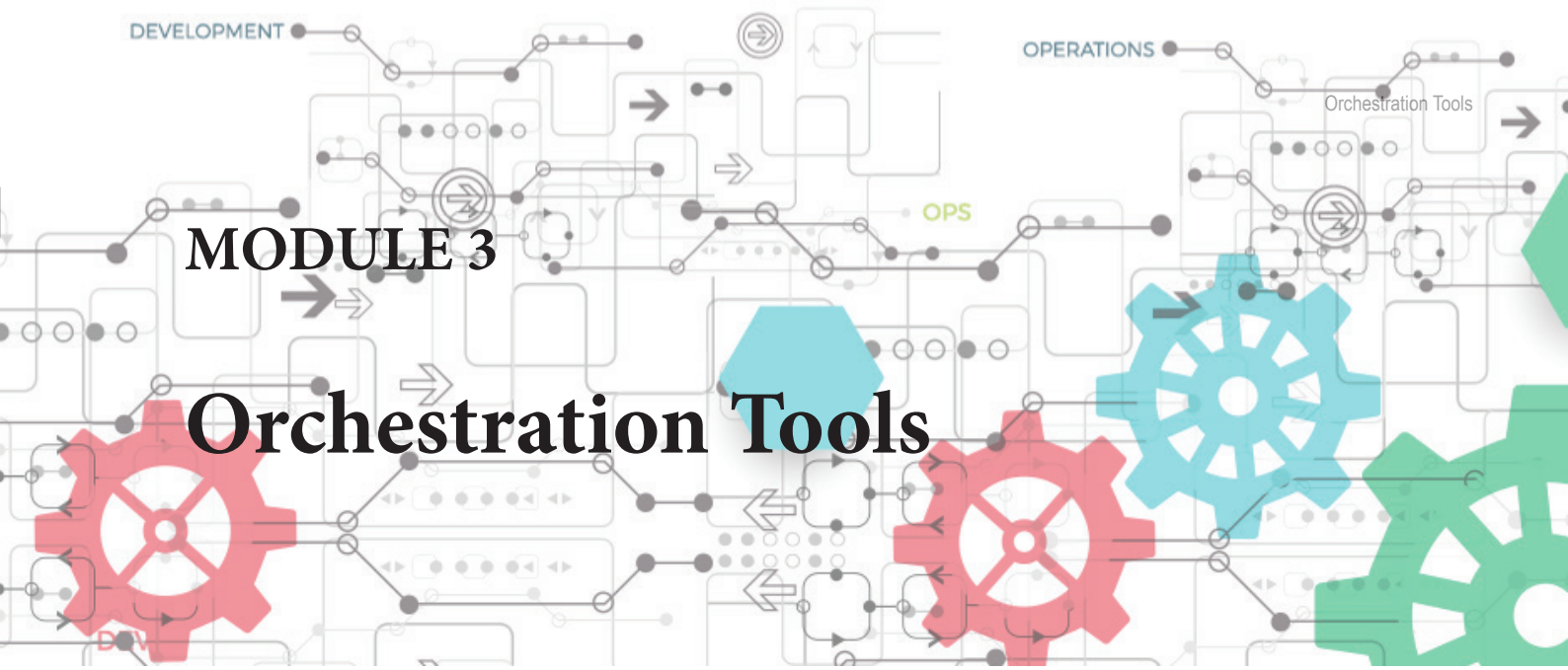
3.15.2 Infrastructure Monitoring 17

3.15.3 Application Performance Monitoring 18

3.16 How to Monitor Containers? 19

3.16.1 Tools to Monitor Containers 20

In a nutshell, we learnt: **20**



MODULE 3

Orchestration Tools

Facilitator Notes:

Welcome the participants and give them an overview of the module. Tell them that they will learn about the 'Orchestration Tools' in this module.

You will learn about the 'Orchestration Tools' in this module.

Module Objectives

At the end of this module, you will be able to:

- Define orchestration and its need
- Explain Docker Swarm and Kubernetes
- Describe AWS (ECS,EKS) and KUBERNETES ON CLOUD
- Discuss the monitoring of containers and how to monitor

Facilitator Notes:

Explain the module objectives to the participants.

You will be informed about the module objectives.

At the end of this module, you will be able to:

- Define orchestration and its need
- Explain Docker Swarm and Kubernetes
- Describe AWS (ECS,EKS) and KUBERNETES ON CLOUD
- Discuss the monitoring of containers and how to monitor

Module Topics

Let us take a quick look at the topics that we will cover in this module:

- What is orchestration?
- Need of orchestration
- Docker Swarm and Kubernetes
- AWS (ECS,EKS)
- KUBERNETES ON CLOUD
- Monitoring of containers
- How to monitor?

Facilitator Notes:

Inform the participants about the topics that they will be learning in this module.

You will learn about the following topics in this module:

- What is orchestration?
- Need of orchestration
- Docker Swarm and Kubernetes
- AWS (ECS,EKS)
- KUBERNETES ON CLOUD
- Monitoring of container
- How to monitor?

3.1 What is Orchestration?

Container orchestration is all about managing the lifecycles of containers, especially in large, dynamic environments.

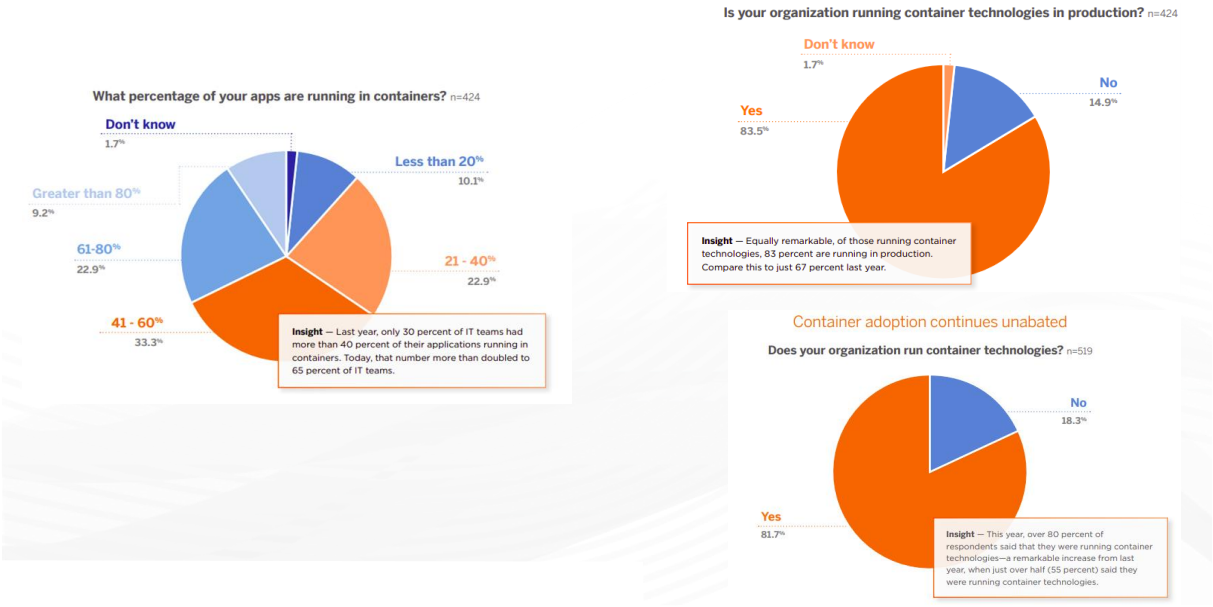
Facilitator Notes:

Explain What is container orchestration.

The orchestration is a process in which a master guides all the musicians in an orchestra to follow a rhythm. And in comparison with containers orchestration means managing the containers with following aspects:

- **Deployment:** This consists deployment of microservices, the way they are supposed to be with respect to the business requirement. This stage consists of building the container image of new versions and pushing them to registry and spinning up container after stopping the previous version of the containers.
- **Scaling:** Since the life cycle of containers is very frequent, which makes them spinning them up and down frequently, this factor helps in scaling up the application and database on a faster pace as compared to standalone applications and databases.
- **Network:** One of the major things which need to be considered while creating an application platform is Network, which is comprised of various factors like exposing the services, load balancing, service discovery, Interservice communication, and so on. Most of the orchestration tools have their inbuilt functionality to simplify these issues.
- **Insight:** As we know containers provides isolation and thus it hides most of the things happening inside them, so we need a robust monitoring setup to keep an eye on how the application and complete stack is behaving. Be it Infra level monitoring, Log Monitoring or Application Monitoring. Orchestration tools have a plugin based mechanism which makes binding the monitoring setup infra less of a hectic task.

3.2 Case study: Need of Orchestration



Source: https://portworx.com/wp-content/uploads/2018/12/Portworx-Container-Adoption-Survey-Report-2018.pdf

Facilitator Notes:

Discuss why container orchestration is important with the help of a survey.

It's clear to most industry watchers that containers have a dramatic impact on the way enterprises deploy applications in the cloud. What isn't obvious is how fast these changes have been happening, why they're taking place, and the profound impact containers are having on new deployment models such as multi-cloud computing. This is perhaps underscored most by the shocking news in October 2018 that IBM is acquiring RedHat, mainly for its OpenShift container platform to strengthen its position in the cloud.

Portworx had released the results of the 2018 Annual Container Adoption survey, which sheds light on these important matters as well as other strategic choices enterprises are making with containers in the cloud. The results are revealing, showing a computing landscape that is being quickly transformed by containers as businesses search for new ways to boost developer productivity, go multi-cloud, reduce infrastructure costs, and respond faster to changing customer needs.

Among the findings, the survey reveals that four out of five enterprises are now running container technologies, and an incredible 83 percent is running them in production — a huge leap from just 67 percent last year. Even we were surprised by the rapid pace at which containers are being put to use for real-world applications.

Equally surprising is the pace at which containers are ushering in new patterns of cloud usage. While increasing developer productivity continues to be the main reason for container adoption, the survey reveals a sharp increase in the proportion of respondents that are deploying containers to enable multi-cloud workloads. Thanks to containers, the promise of multi cloud — with its benefits of improving uptime, avoiding lock-in and gaining price leverage — is fast becoming a reality.

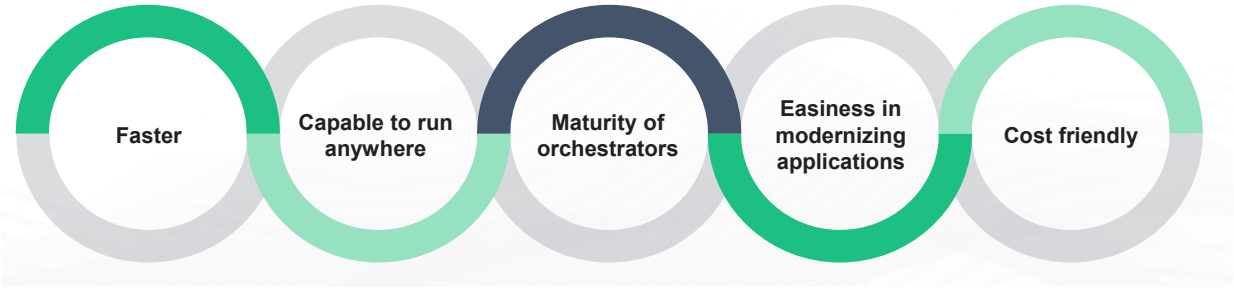
The survey also dives deep into questions about developer preferences and perceptions of containers in the cloud. Which public cloud is the most popular for running containers? Which is the most developer-friendly for container use? What's the most widely used container scheduler? The survey has covered all these questions and more, and some of the answers may surprise you.

The survey explores the state of container technologies across a wide variety of industries and company sizes, to build a full picture of the current state of container usage. To know more about the results of 2018 Annual Container Adoption Survey, check out the link below:

https://portworx.com/wp-content/uploads/2018/12/Portworx-Container-Adoption-Survey-Report-2018.pdf

3.2.1 Need of Orchestration: Container and Microservices

Some reasons behind the need of orchestration are as follows:



Facilitator Notes:

Discuss with the participants why Microservices are the next big thing.

By far now we have discussed that why we need orchestration to manage containers, but the question still persists is that why everyone is moving to Microservice architecture and majorly containers. So there are some reasons behind that.

- **Faster:** Containers are designed to be leaner, faster and more reliable than VMs. Containers are nothing more than virtualized processes that live indefinitely, or live for as long as they are needed, which means that they offer all the functionality of an application running inside a virtual machine, but without the overhead. They are a lot smaller than a VM (~100 MB vs. 4 GB), they take ~1s to start vs. 30s, and are less likely to crash since all the dependencies that enable them to run come packaged in something called an image (more on this later). This means that on the same compute capacity and memory of 1 VM you could run 5+ containers.
- **Capable to run anywhere:** Containers run on an all hardware platforms, from Raspberry Pi, to mainframes. Windows is relatively new to the game, but getting lots of traction due to its broad following from GUI-loving admins to console-only geeks – thanks to PowerShell. Why? Because Windows runs on the most popular architecture of the enterprise: x86/x64.
- **Maturity of orchestrators:** The two most popular container orchestration technologies (Kubernetes and Docker Swarm) are capable of managing Windows containers now. To show you how serious Microsoft is about containers, not only have they become members of the Cloud Native Computing Foundation (CNCF), but they curated the latest release of Kubernetes (1.8), formerly a Google project which has been donated to the Open-source community.
- **Easiness in modernizing applications:** You can modernize apps with little to no-effort using containers. Front-end web servers (such as IIS) can run in a container. Traditional Windows apps (.Net 3.5, 4) as well as modern apps (.Net Core) can run in containers with virtually no code change. SQL server can also run inside a container since the release of SQL 2017. This means that you can modernize a 3-tier app a lot faster than you thought.
- **Cost friendly:** Containers are portable and do not lock-you-in. You can run containers on IaaS in Azure, AWS or GCP, on spare capacity that you may have on your on-prem Hypervisors, or on any VM that runs a modern OS like Windows 2016 or Linux.

3.3 Orchestration Tools

The various container orchestration tools available in the market are as follows:

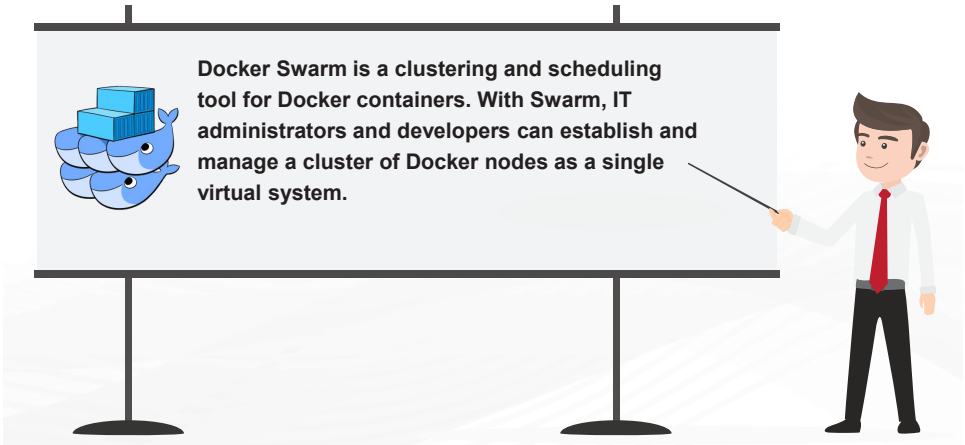
Facilitator Notes:

Inform the participants about the various container orchestration tools available in the market.

The various container orchestration tools available in the market are as follows:

- Docker-swarm
- Azure Container Services
- Cloud Foundry Diego
- Mesosphere Marathon
- Amazon ECS
- CoreOS Fleet
- Kubernetes
- Google Container Engine

3.4 Docker Swarm



Facilitator Notes:

Explain Docker Swarm.

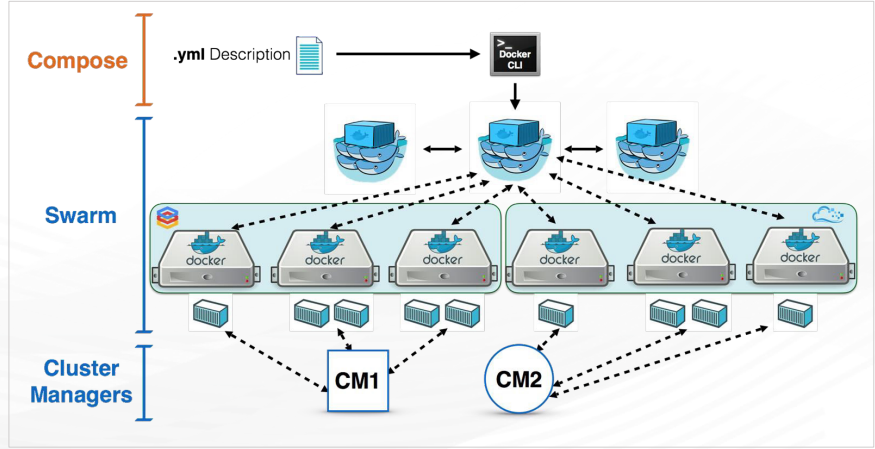
Docker-swarm: Docker's native orchestration platform also called as simply swarm. Since, it is the docker's own orchestration platform, it came with lots of pros, but as other enterprises also kept developing their platforms religiously. It began lagging behind in a couple of factors.

This orchestration doesn't required any other tool other then docker swarm, all of the nodes should have docker installed and swarm mode activated. Since orchestration always work among multiple nodes we need to assign particular nodes to be master and worker node.

Swarm is being used where the infra is not much typical and exposing services in not much of a hassle, but things change when the infra becomes more typical and we need more functionalities to manage containers then other advanced orchestration tools take place.

3.5 Docker Swarm Architecture

Let's look at the Docker swarm architecture.



*Source: https://miro.medium.com/max/3774/1*dvIZd0ZN-KcjaDDAKTWcLA.png*

Facilitator Notes:

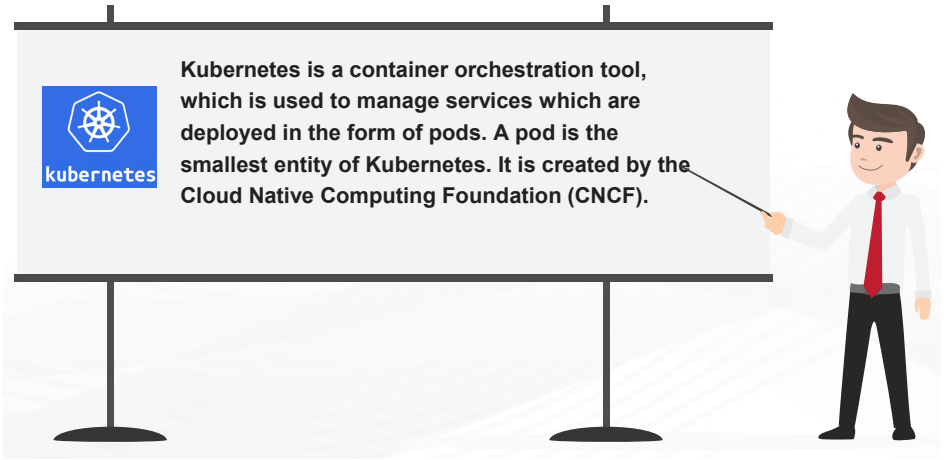
Explain Docker Swarm architecture.

Docker Swarm has three key components Docker Node, Docker Services, and Docker tasks in its architecture.

- **Docker Node:** Docker Swarm runs in a cluster, hence it has multiple nodes classified as:
  - **Manager Node:** Responsible for all orchestration and container management tasks required to maintain the system in the desired state, such as scheduling the services, maintaining the cluster state, and servicing the swarm mode HTTP endpoints.
  - **Worker Node:** All kinds of services are deployed and executed on these nodes, they carry out all the heavy tasks asked by Manager nodes. Worker nodes are also instances of Docker Engine, whose sole purpose is to run containers. Worker nodes don't participate in taking the decision as the master does, or serve the swarm mode HTTP API.

- **Service and tasks:** A service is the definition of the tasks to execute on the manager or worker nodes. It is the central structure of the swarm system and the primary root of user interaction with the swarm.

3.6 Kubernetes



Facilitator Notes:

Inform the participants about Kubernetes.

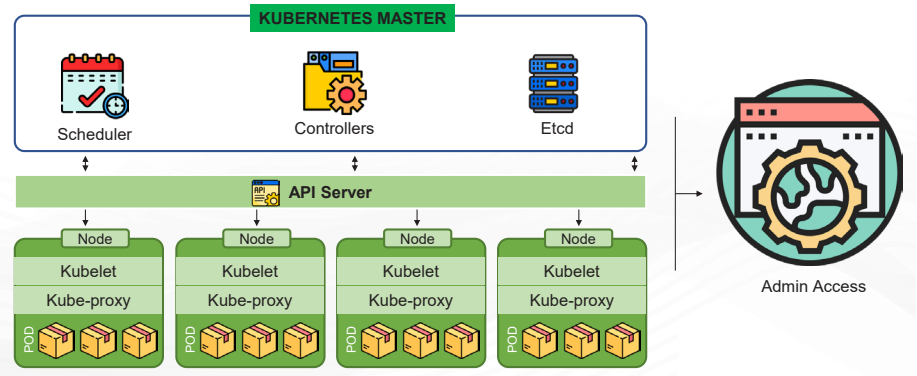
Kubernetes is a container orchestration tool, which is used to manage services which are deployed in the form of pods. A pod is the smallest entity of kubernetes. It is created by the Cloud Native Computing Foundation (CNCF).

Kubernetes has various features like:

- Infrastructure in containerized form
- Allows flexibility for having continuous integration, continuous development, and continuous deployment
- It empowers admins to have a proper resource utilization
- Creating multiple environment is easy because of isolation in terms of Namespaces
- Load balancing is a very easy concept, and hence auto-scaling of entire infrastructure is not a hassle
- It has an application oriented approach

3.6.1 Kubernetes Architecture

Kubernetes works on client-server architecture and for high availability, it has it works on multi node.



Facilitator Notes:

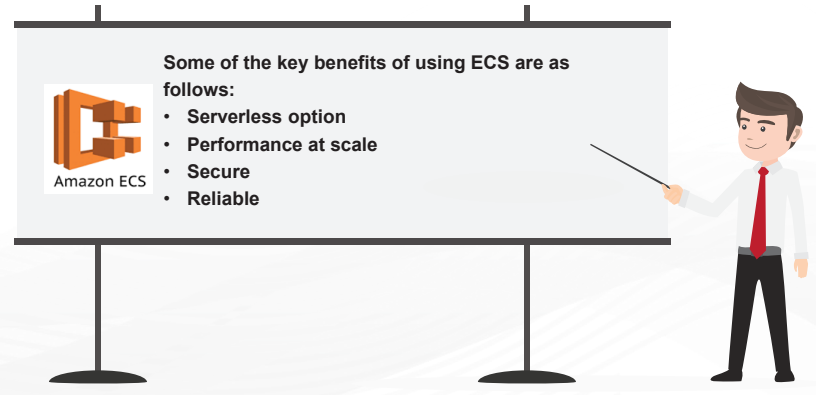
Describe the Kubernetes architecture.

Kubernetes works on client-server architecture and for high availability, it has it works on multi node. There are two types of nodes **Master** Node & **Minion** Node.

Kubernetes has many components to ensure the site availability 24x7. Let's discuss about them.

- **Master Components:** They provide cluster control plane by taking global decision like scheduling, delegation and responding to cluster events.
  - **Kube API server:** It is a front-end for kubernetes control plane. It scales horizontally deploying more instances.
  - **Etcd:** It is a highly available and consistent key value store used as kubernetes backing store for all cluster data.
  - **Kube scheduler:** It watches newly created pods, that have no node assigned and selects a node for them to run on.
  - **Kube control manager:** It is a daemon that embeds the core control loops shipped with kubernetes.
- **Node Components:** These nodes do all the heavy lifting of the cluster like running services and pods, the actual application pods run over these nodes.
  - **Kubelet:** It is an agent that runs on each node in the cluster. It makes sure that containers are running in a pod.
  - **Kube proxy:** It enables the kubernetes service abstraction by maintaining network rules on the host and performing connection forwarding.
  - **Container runtime:** It is a container execution environment, which ensures the allocation of limited shares of resources (e.g., CPU, memory, disk) to the containerized application, also exposes the services and APIs and tools for managing containers. It acts as a layer between the host machine and the containers fulfilling all sorts of requirements like port and volume mapping needed to achieve containerized services. For example: LXD, Docker daemon, Rkt process.

3.7 Amazon Web Services Elastic Container Services



Facilitator Notes:

Explain the Elastic Compute Service.

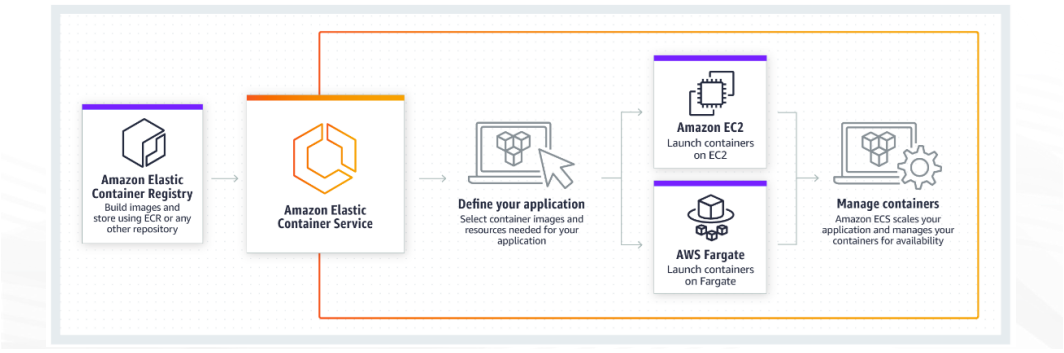
Elastic Compute Service is provided by Amazon web services to host containerised applications over cloud and orchestrate them as ECS Service. Unlike any other orchestration tool, it doesn't have control plane from a management perspective, but internally has its own architecture.

The key benefits of using ECS are as follows:

- **Serverless option:** Since ECS supports fargate, hence you don't need to worry about the server provisioning. It lets you specify and pay for resources per application, and improves security through application isolation by design.
- **Performance at scale:** You can launch thousands of containers with ECS with much less complexity.
- **Secure:** It lets you launch containers within VPC, which is quite reliable in terms of security. You can also assign granular access permissions for each of your containers using IAM to restrict access to each service and what resources a container can access.
- **Reliable:** Since you are using it as a service, so there are very less chances of encountering failure from AWS end.

3.8 AWS Elastic Container Services Architecture

Let's understand about the architecture of an elastic compute service.



Source: https://d1.awsstatic.com/diagrams/product-page-diagrams/product-page-diagram_ECS_1.86ebd8c223ec-8b55aa1903c423fbe4e672f3daf7.png

Facilitator Notes:

Inform the participants about the architecture of an elastic compute service.

An elastic compute service has several components, since it is a service, the major emphasis is on to provide input to service and the rest will be taken care by it.

- **Cluster:** It is the group of servers where your container services will be running. AWS provides two options to have cluster; EC2 and Fargate.
  - **EC2:** It is the standard service in which we will provision our cluster using EC2 instances and all sorts of management will be taken care by us like Autoscaling, etc.
  - **Fargate:** It is a serverless platform on which your container services will be running, in this case you don't have to worry about server provisioning.
- **ECR(Elastic Compute Registry):** As we know to run containers we need docker images, and in container lifecycle it is better to have an in-house hosted container registry. AWS provides that container registry in the form of an Elastic Compute Registry.
- **Task Definition:** If we consider a service comprises of multiple containers, then the task definition is json template which consists of all the meta information about the containers.
  - **Container Definition:** It is the part of task definition in which ports, volume, logging, limits, etc., are defined, the parameters which we mention in docker run command.
- **Services:** We create services using a task definition. It means we have defined that how we want our containers to run in task definition and to execute that definition, we create a service in which we mention multiple parameters like service discovery, load balancing, autoscaling and in which cluster it is to be launched.

3.9 AWS Elastic Kubernetes Services(EKS)

Facilitator Notes:

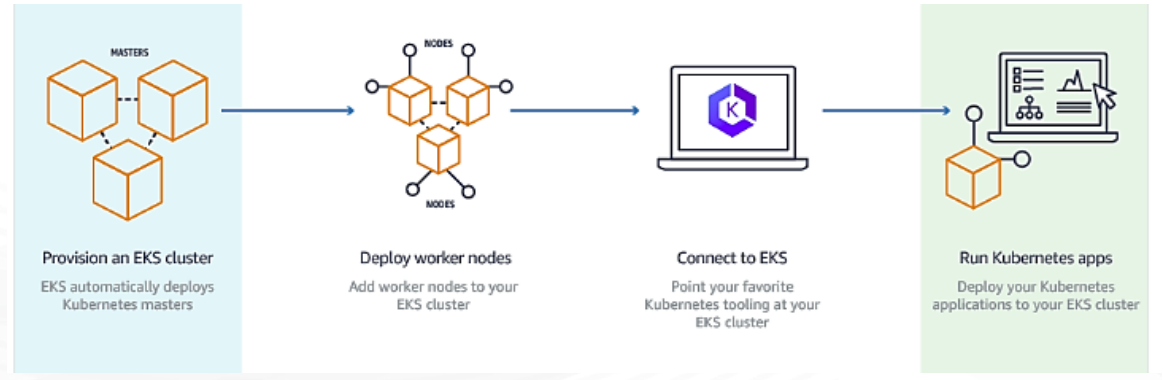
Discuss the Amazon Elastic Kubernetes Service.

Amazon Elastic Kubernetes Service: As the name implies, it is a kubernetes as a service provided by an Amazon web services. It claims that it enables you to run containers in production by provisioning VM and clusters in your cloud infrastructure. It is an orchestrator agnostic platform which makes running containers as per your business needs.

The main highlight of EKS is we don't need to stand up or maintain our own Kubernetes control plane, since it is managed by the service itself.

3.9.1 AWS Elastic Kubernetes Services(EKS) Architecture

Let's understand about the architecture of Amazon Elastic Kubernetes Service.



Source: https://docs.aws.amazon.com/eks/latest/userguide/images/what-is-eks.png

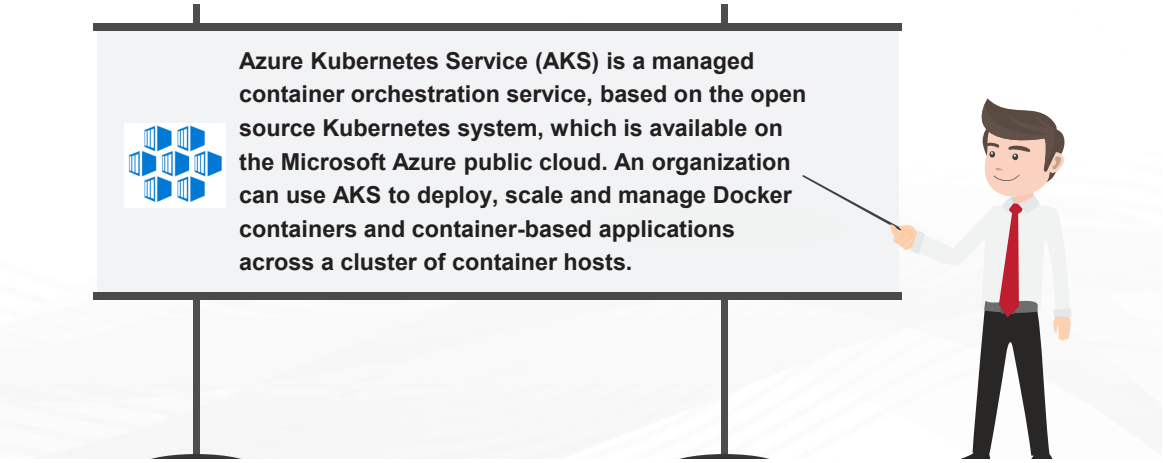
Facilitator Notes:

Explain the architecture of Amazon Elastic Kubernetes Service.

There is a bit of similarity in ECS and EKS which is that both of the services provide solution for hosting the application on a dockerized platform. The difference arises when we dig into the backend technology being used, in ECS its AWS's own orchestration tool and in EKS it uses kubernetes.

The components in its architecture are same as of Kubernetes, the only advantage is that we are not supposed to manage any control plane components.

3.10 Azure Kubernetes Services



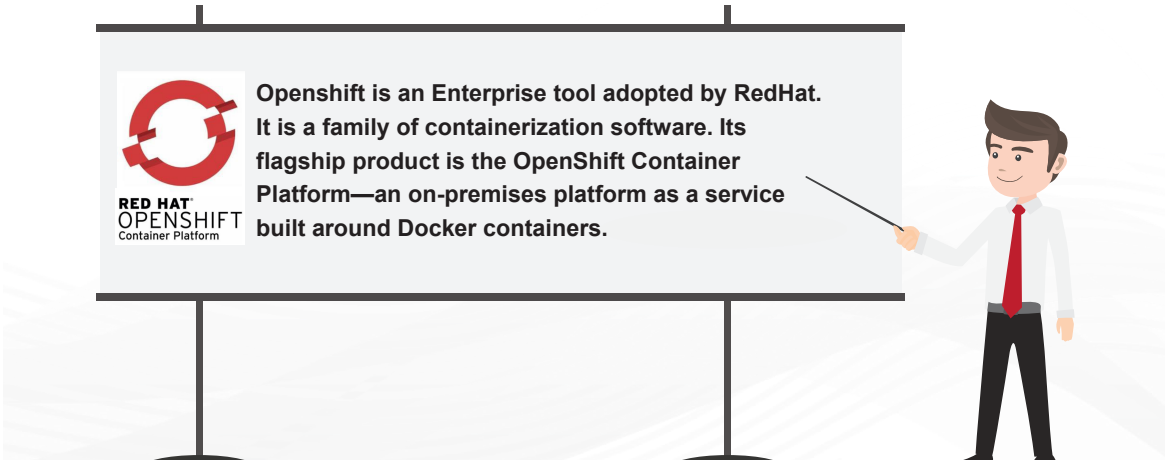
Facilitator Notes:

Describe the participants about the Azure container services.

Azure Container Services: As the name implies, it is a container service provided by a cloud platform named Microsoft Azure, which supports many open source tools for containers and their orchestration. It claims that it enables you to run containers in production by provisioning VM and clusters in your cloud infrastructure. It is an orchestrator agnostic platform which makes running containers as per your business needs.

Azure Kubernetes Service (AKS) is a managed container orchestration service, based on the open source Kubernetes system, which is available on the Microsoft Azure public cloud. An organization can use AKS to deploy, scale and manage Docker containers and container-based applications across a cluster of container hosts.

3.11 Openshift



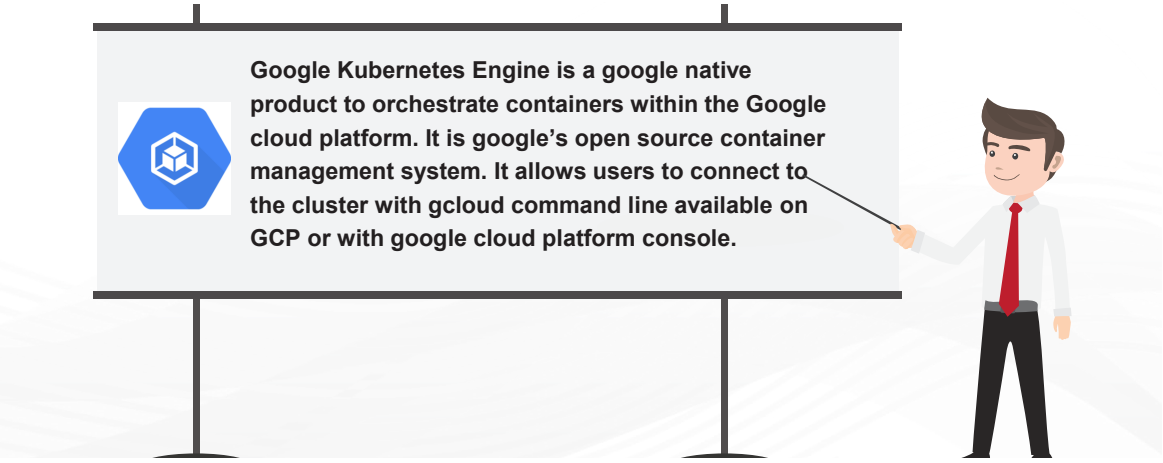
Facilitator Notes:

Explain the participants about openshift.

Openshift is an Enterprise tool adopted by RedHat. It is a family of containerization software. Its flagship product is the OpenShift Container Platform—an on-premises platform as a service built around Docker containers. It is basically a wrapper around orchestrator called Kubernetes on a foundation of Red Hat Enterprise Linux.

OpenShift Container Platform is a private platform as a service (PaaS) for IT software companies which are looking for deploying and orchestrating containers on their own on-premises hardware or on the infrastructure of a cloud provider.

3.12 Google Kubernetes Engine



Facilitator Notes:

Inform the participants about Google kubernetes engine.

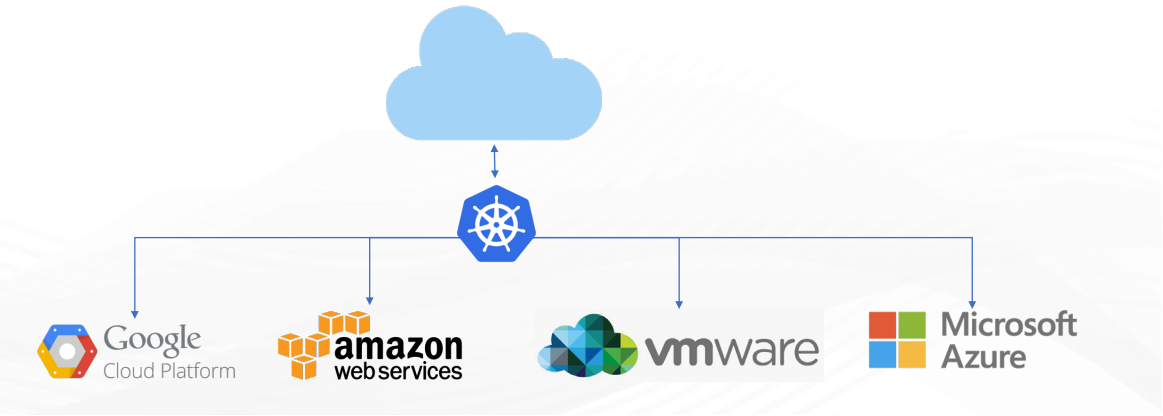
Google Kubernetes Engine is a google native product to orchestrate containers within the Google cloud platform. It is google's open source container management system. It allows users to connect to the cluster with gcloud command line available on GCP or with google cloud platform console.

There are various operation, which can be done on GKE. Some of them are as follows:

- Building images, creating or resizing Docker container clusters
- Creation of container pods, replication controllers, jobs, services or load balancers all of the operations which are part of orchestrating containers as pods
- Manipulating the application controllers
- Update/upgrade or downgrade the container clusters functioning inside the GCP platform
- Debugging of container clusters inside the GKE

3.13 Kubernetes on cloud

Here, we will learn about kubernetes on cloud.



Facilitator Notes:

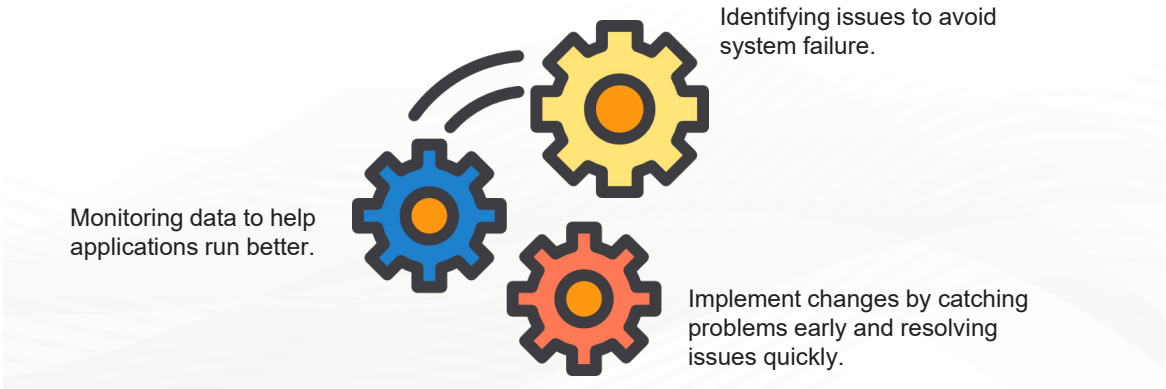
Explain about the Kubernetes on cloud.

Kubernetes is flexible enough that you can couple cloud services with it. There are many cloud providers which are compatible with Kubernetes. Some of them are as follows:

- AWS
- Azure
- CloudStack
- GCE
- OpenStack
- OVirt
- Photon
- vSphere
- IBM Cloud Kubernetes Service
- Baidu Cloud Container Engine
- Tencent Kubernetes Engine

3.14 Why we Need to Monitor Containers

We need to monitor containers because of the following reasons:



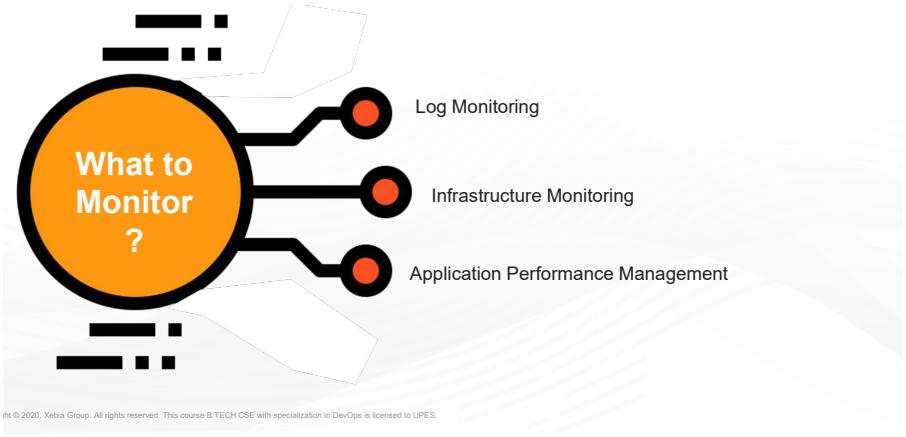
Facilitator Notes:

Discuss with the participants why we need to monitor containers.

Let's talk about monitoring keeping container aside. Monitoring is one of the crucial aspects of IT infrastructure. Typically, in the software industry, we develop an application, test it thoroughly in multiple environments, and then make it live in production. But, things don't end here, if we want to ensure the availability of site 24x7, we need a way to assess or monitor that our application is responding properly, not creating any lag and many things are to be taken care of in that aspect.

Containers tendency to provide isolation becomes a glitch while keeping an eye on it to monitor. Because in case of VMs and bare metal, if we are using any tool, it will read the filesystem of the machine on which the application is running, but it in case of container nothing is shared on the host machine regarding the processes running inside it. Hence, the big challenge is to retrieve information about the metrics without hampering the application.

3.15 What Needs to be Monitored in Containers?



Facilitator Notes:

Explain what needs to be monitored in containers.

There are several aspects of any system which you need to pay attention in the software industry. This comprises of everything like status of the platform, where your application is running, events occurring in your system, and how your system is performing. If you classify these into a few categories, it will be like:

- Log Monitoring
- Infrastructure Monitoring
- Application Performance Management

We will discuss about these categories in the next slide.

3.15.1 Log Monitoring

Logs are the events which occur in a system when an application gets used by another resource, or when it computes the logic.

Facilitator Notes:

Describe what is log monitoring.

Logs are the events which occur in a system when an application gets used by another resource, or when it computes the logic. These logs are generally appended in a file on a system. Application logs are the events occurred when a user accesses the application, which are appended in a file or stdout in a system depending upon the logging configuration of an application.

Monitoring of logs means that we want to keep a close eye on the events of the application if any unexpected event occurs. There are several ways to do it, the most basic one is to read the file in which the logs are appended, but if we are dealing with multiple applications or multiple microservices, it will be a problem to log into the servers and lookup into a log file. To have a better approach, organisations decided to move to a solution which will provide them the logs in a centralised location with a better visibility.

Imagine a user interface where you can select the service and it will show you the run time logs, even the previous ones when you select the different time frame. Most of the monitoring solutions tend to provide such ease.

3.15.2 Infrastructure Monitoring

When we talk about systems in our infrastructure, it literally translates to the servers, be it Unix based or Windows based. It is the platform where our applications are running and consuming the resources of our servers. Some of the resources which are used by a system are CPU, RAM, Storage and network.

Facilitator Notes:

Explain what is infrastructure monitoring.

When we talk about systems in our infrastructure, it literally translates to the servers, be it Unix based or Windows based. It is the platform where our applications are running and consuming the resources of our servers. Some of the resources which are used by a system are CPU, RAM, Storage and network.

Suppose a scenario when couple of applications are running on a server, and suddenly there is a spike in the resource utilisation due to increase in number of hits, but you don't know about it because there is no mechanism in place which will tell you about such severity.

There arises the importance of Infrastructure Monitoring, we need a mechanism in place which could assess our infrastructure and going forward it should inform us about such criticality.

3.15.3 Application Performance Monitoring

Application Performance Monitoring (APM) is an essential tool to monitor the performance of an application. There several parameters on which an application can be monitored.

- App metrics
- Code level performance
- Network based

Facilitator Notes:

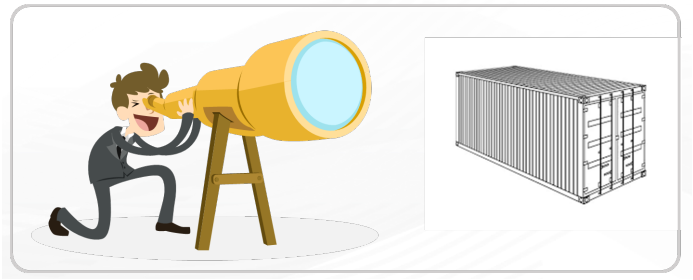
Discuss application performance monitoring.

Application Performance Monitoring (APM) is an essential tool to monitor the performance of an application. There several parameters on which an application can be monitored.

- **App metrics:** These are the metrics which tells how many requests are being held by the application and shows which URLs are functioning slow. Several tools use various server and app metrics and call it APM. Since, they don't go deep into code level profiling, they can't tell you why the performance is slow.
- **Code level performance:** There are a few tools which provide code level profiling like Stackify Retrace, New Relic, AppDynamics, and Dynatrace. These are the typical types of APM products. They also monitor the transaction tracing of the application.
- **Network based:** The term used by APM tools in regards with the network is called as Extrahop. It specifically looks after the performance of the network calls made by the application and if it does the profiling,it will tell you the reason as well. Such tools fall under the category called NPM that focuses on this type of solutions.

3.16 How to Monitor Containers?

Monitoring of containers won't be a hustle if we understand the source, shipper, aggregator required to monitor an Infra.



Facilitator Notes:

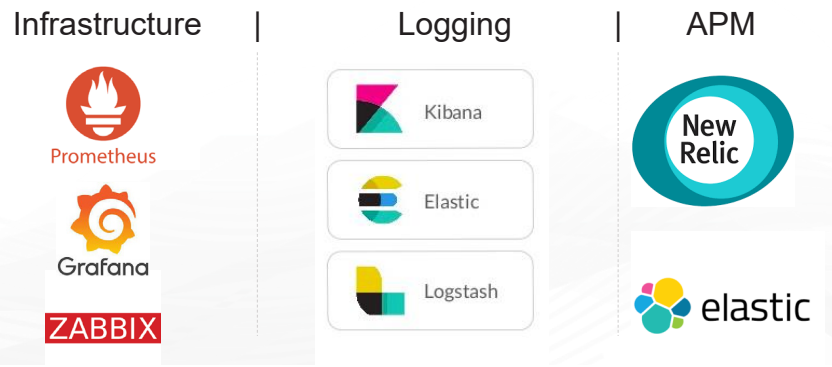
Explain the participants how to monitor containers.

We have discussed various aspects on which we can monitor the containers, Now, let's discuss how to do it. Any utility which is supposed to monitor the system will get the data from the system in the form of **metrics.** These metrics are nothing but the data which is being written somewhere on the file system.

Breaking down the HOW of monitoring we need to look at the sequence of how it will be done:

- **Source:** It is the point where the metrics are being generated, for example: If you want to monitor CPU resources you will read the file system on the server where this information is being appended, somewhere in /proc if linux is concerned. So, the utility will ultimately read this file system to get the metrics.
- **Shipper:** Depending upon the architecture of the monitoring system, you will need these metrics to be shipped from the server to that portal where it will be interpreted.
- **Aggregator:** It could be called as a collector where all those metrics will be stored in some sort of database from where you can further use them for monitoring, visualising and alerting.
- **Visualisation:** It is the part where you can see what is happening in the system in human understandable form like graphs, pie chart, etc. Here, you will only be looking at metrics in centrally visualised form, you don't need to login to the servers for the metrics.
- **Alerting:** Without alerting monitoring just a setup where you can see what is happening and you need a person, who will constantly look at the dashboards for any issue, which sounds inappropriate use of human resources. To get alerts on mediums like mail, phone, slack, etc., you will set the threshold on certain values which, when surpasses, you will get notified on any of the above mentioned channels.

3.16.1 Tools to Monitor Containers



Source: https://image.slidesharecdn.com/elkwebinar-internal-161020195048/95/search-analytics-with-elk-elastic-stack-17-638.jpg?cb=1477055974

Source: Prometheus, grafana, alertmanager logo from google

Facilitator Notes:

In this sub-module you will explain about various tools in market to monitor containers?

In this you will introduce multiple tools present in market to monitor the containers.

- Infrastructure Monitoring: Zabbix, Prometheus, Grafana, TICK stack, etc.
- Log Monitoring: ELK, EFK, Graylog, Datadog, Sumologic, etc.
- APM: New Relic, ELK 7.0, Glowroot etc.

In a nutshell, we learnt:

1. What is orchestration?
2. Need of orchestration
3. Docker Swarm and Kubernetes
4. AWS (ECS,EKS)
5. KUBERNETES ON CLOUD
6. Monitoring of containers
7. How to monitor?

Facilitator Notes:

Share the module summary with the audience.

Ask the participants if they have any questions. They can ask their queries by raising their hands.

Now, you have reached the end of the module, In this module, you have learned:

- What is orchestration?
- Need of orchestration
- Docker Swarm and Kubernetes
- AWS (ECS,EKS)
- KUBERNETES ON CLOUD
- Monitoring of containers
- How to monitor?

Release Notes

B. TECH CSE with Specialization in DevOps

Semester Six -Year 03

Release Components.
Facilitator Guide, Facilitator Course Presentations, Student Guide, Mock exams and relevant lab guide.

Current Release Version.
1.0.0

Current Release Date.
19 Jan 2020

Course Description.
Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Jan 2021