**Week12**

**Container**

[Containers](https://www.ibm.com/topics/containers) are lightweight, executable application components that combine application source code with all the operating system (OS) libraries and dependencies required to run the code in any environment.

**Orchestration**

Orchestration is the automated configuration, management, and coordination of computer systems, applications, and services. Orchestration helps IT to more easily manage complex tasks and workflows.

**Container orchestration**

Container orchestration automates the provisioning, deployment, networking, scaling, availability, and lifecycle management of containers.

[Kubernetes](https://www.ibm.com/topics/kubernetes) is the most popular container orchestration platform, and most leading public cloud providers - including Amazon Web Services (AWS), Google Cloud Platform, IBM Cloud and Microsoft Azure - offer managed Kubernetes services.

Other container orchestration tools include Docker Swarm and Apache Mesos.

**What is container orchestration engine?**

Container orchestration is the automation of much of the operational effort required to run containerized workloads and services. This includes a wide range of things software teams need to manage a container's lifecycle, including provisioning, deployment, scaling (up and down), networking, load balancing and more.

**Orchestration tools**

1. Kubernetes

2. Openshift

3. Nomad

4.Docker Swarm

5. Docker Compose

6. MiniKube

7. Marathon

8. Cloudify

**Kubernetes**

Kubernetes — also known as “k8s” or “kube” — is a container orchestration platform for scheduling and automating the deployment, management, and scaling of containerized applications.

**Why Kubernetes?**

Service discovery and load balancing:  If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.

Storage orchestration: allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more

Automated rollouts and rollbacks:  it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.

Automatic bin packing: You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.

Self-healing: Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

Secret and configuration management: Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

**Kubernetes configuration**

- **Deployment**

A Deployment provides declarative updates for [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) and [ReplicaSets](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/" \t "_blank).

You describe a desired state in a Deployment, and the Deployment [Controller](https://kubernetes.io/docs/concepts/architecture/controller/) changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

**- Service**

In Kubernetes, a Service is a method for exposing a network application that is running as one or more [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) in your cluster.

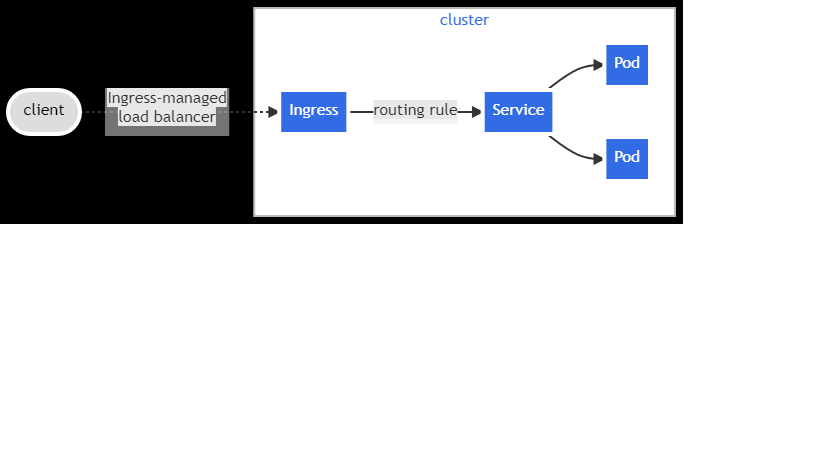
A key aim of Services in Kubernetes is that you don't need to modify your existing application to use an unfamiliar service discovery mechanism. You can run code in Pods, whether this is a code designed for a cloud-native world, or an older app you've containerized. You use a Service to make that set of Pods available on the network so that clients can interact with it.

**- Load balancer/ingress**

## What is Ingress?

[Ingress](https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.28/#ingress-v1-networking-k8s-io) exposes HTTP and HTTPS routes from outside the cluster to [services](https://kubernetes.io/docs/concepts/services-networking/service/) within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

Here is a simple example where an Ingress sends all its traffic to one Service:

****

Disaster recovery

Disaster recovery is the process by which an organization anticipates and addresses technology-related disasters.

[Kubernetes disaster recovery](https://www.rubrik.com/solutions/kubernetes) can be broken down into two phases: backup and recovery. Backup is the process of preserving data before any disaster strikes, while recovery entails getting back up after one has occurred.

**Deployment Strategies to Consider**

Deployment strategies are practices used to change or upgrade a running instance of an application. The following sections will explain six deployment strategies. Let’s start with discussing the basic deployment.

Blue/green deployment is a deployment technique to release new code into the production environment. Blue/green deployments make use of two identical production environments — one of these is active and the other environment is set to idle. New updates are pushed to the active environment where it is monitored for bugs while the idle environment serves as a backup where traffic can be routed in case an error occurs.

Blue/green deployment provide the following benefits to businesses

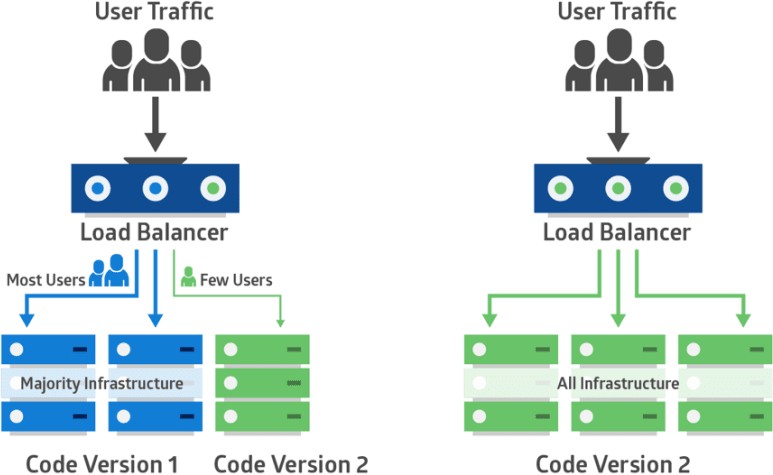
* + Simple, fast, easy to understand and implement.
  + Rollback is straightforward as teams need to simply flip traffic back to the old environment in case any issue arises.
  + Blue-green deployments are not as risky and vulnerable to losses as compared to other deployment strategies.



Canary deployment is a technique to reduce the risk of updating software or introducing new changes in the production environment by slowly rolling out the change to a small subset of users before making the software functional for everyone.

Canary deployments provide the following benefits to businesses.

* + Allows enterprises to test in production with real users and use cases.
  + Enables comparison of different service versions side by side.
  + Cheaper than blue-green deployments because it does not require two production environments.
  + DevOps team can rapidly and safely trigger a rollback to a previous version of an application.



**What is Disaster Recovery?**

Disaster recovery is the process by which an organization anticipates and addresses technology-related disasters. The process of preparing for and recovering from any event that prevents a workload or system from fulfilling its business objectives in its primary deployed location, such as power outages, natural events, or security issues.

**Disaster recovery and their types:**

1. Data Center Disaster Recovery

2. Network Disaster Recovery

3. Virtualized Disaster Recovery

4. Disaster Recovery in the Cloud

5. Disaster Recovery as a Service (DRaaS)

**Why is disaster recovery important?**

A disaster is an unexpected problem resulting in a slowdown, interruption, or network outage in an IT system. Outages come in many forms, including the following examples:

* An earthquake or fire
* Technology failures
* System incompatibilities
* Simple human error
* Intentional unauthorized access by third parties
* Ensures business continuity
* Enhances system security
* Improves customer retention
* Reduces recovery costs

**How does disaster recovery work?**

Disaster recovery focuses on getting applications up and running within minutes of an outage. Organizations address the following three components.

* Prevention

To reduce the likelihood of a technology-related disaster, businesses need a plan to ensure that all key systems are as reliable and secure as possible. Because humans cannot control a natural disaster, prevention only applies to network problems, security risks, and human errors. You must set up the right tools and techniques to prevent disaster. For example, system-testing software that auto-checks all new configuration files before applying them can prevent configuration mistakes and failures.

* Anticipation

Anticipation includes predicting possible future disasters, knowing the consequences, and planning appropriate disaster recovery procedures. It is challenging to predict what can happen, but you can come up with a disaster recovery solution with knowledge from previous situations and analysis. For example, backing up all critical business data to the cloud in anticipation of future hardware failure of on-premises devices is a pragmatic approach to data management.

* Mitigation

Mitigation is how a business responds after a disaster scenario. A mitigation strategy aims to reduce the negative impact on normal business procedures. All key stakeholders know what to do in the event of a disaster, including the following steps.

* Updating documentation
* Conducting regular disaster recovery testing
* Identifying manual operating procedures in the event of an outage
* Coordinating a disaster recovery strategy with corresponding personnel



**What are the key elements of a disaster recovery plan?**

An effective disaster recovery plan includes the following key elements.

Internal and external communication

The team responsible for creating, implementing, and managing the disaster recovery plan must communicate with each other about their roles and responsibilities. If a disaster happens, the team should know who is responsible for what and how to communicate with employees, customers, and each other.

Recovery timeline

The disaster recovery team must decide on goals and time frames for when systems should be back to normal operations after a disaster. Some industries’ timelines may be longer than others, while others need to be back to normal in a matter of minutes.

The timeline should address the following two objectives.

Recovery time objective

The recovery time objective (RTO) is a metric that determines the maximum amount of time that passes before you complete disaster recovery. Your RTOs may vary depending on impacted IT infrastructure and systems.

Recovery point objective

A recovery point objective (RPO) is the maximum amount of time acceptable for data loss after a disaster. For example, if your RPO is minutes or hours, you will have to back up your data constantly to mirror sites instead of just once at the end of the day.

Data backups

The disaster recovery plan determines how you back up your data. Options include cloud storage, vendor-supported backups, and internal offsite data backups. To account for natural disaster events, backups should not be onsite. The team should determine who will back up the data, what information will be backed up, and how to implement the system.

Testing and optimization

You must test your disaster recovery plan at least once or twice per year. You can document and fix any gaps that you identify in these tests. Similarly, you should update all security and data protection strategies frequently to prevent inadvertent unauthorized access.

**What is load balancing?**

Load balancing is the method of distributing network traffic equally across a pool of resources that support an application. Modern applications must process millions of users simultaneously and return the correct text, videos, images, and other data to each user in a fast and reliable manner. To handle such high volumes of traffic, most applications have many resource servers with duplicate data between them. A load balancer is a device that sits between the user and the server group and acts as an invisible facilitator, ensuring that all resource servers are used equally.

Load balancers improve application performance by increasing response time and reducing network latency.

**They perform several functions such as the following:**

Distribute the load evenly between servers to improve application performance

Redirect client requests to a geographically closer server to reduce latency

Ensure the reliability and performance of physical and virtual computing resources

**What is application monitoring?**

Application monitoring is the process of measuring application performance, availability and user experience and using this data to identify and resolve application issues before they impact customers. Application monitoring is difficult due to the dynamic nature of today’s hybrid cloud and cloud native environments. The most effective modern approaches incorporate full stack monitoring from the front-end, user experience, to the back-end infrastructure to deliver complete visibility into application performance.

**How Application Performance Monitoring Works**

APM leverages advances in web server hardware, "big data" architecture, and load balancing on [cloud networks](https://www.vmware.com/topics/glossary/content/cloud-networking.html" \t "_self) to get detailed analytics of software code that is in production. There are many APM tools available on the market. The right one for your organization will depend on your specific needs. Some key features may include robust monitoring tools, customizable and user-friendly dashboards, alerting mechanisms, and accessible visualizations tools for critical metrics.

 Application performance monitoring is based on real-time I/O packet scanning of data transmissions on a network. Analyzing and correlating the data makes it possible to detect performance issues and diagnose problems. APM solutions typically provide a dashboard that gives visibility into the performance of an application and may display data like response time, error rate, and throughput. Event logs can be used to debug problematic software code or database queries. And network diagnostics assist with resource planning, hardware allocation, and elastic web server automation to achieve greater cost efficiency.

**APM Dependencies**

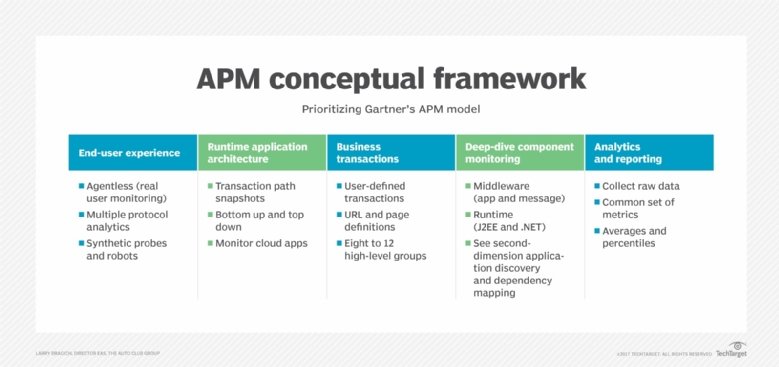
As part of the application monitoring process, APM teams typically map out all dependencies and use the application performance monitoring tools to also track the performance of the various dependencies in real-time. Because dependencies directly affect the performance of apps, it’s crucial to pay attention to them. Common dependencies include SQL databases, caching solutions, message queues, and HTTP APIs.

**What are the components of APM?**

Application performance monitoring focuses on tracking five primary components of app performance:

* runtime application architecture
* [real user monitoring](https://www.techtarget.com/searchitoperations/definition/real-user-monitoring-RUM)
* business transactions
* component monitoring
* analytics and reporting

Runtime application architecture. This aspect analyzes the hardware and software components used in the app's execution and the paths they use to communicate. Through pattern recognition and the identification of performance problems, IT professionals can anticipate the potential for future problems before they happen and plan for necessary upgrades -- such as more application storage -- in a timely manner.



**These components make up the APM conceptual framework.**

Real user monitoring. Also known as end user experience monitoring, this component gathers user-based performance data to understand how well the application is performing for users and to gauge potential performance problems. For example, APM might monitor the response time of a critical website and flag response times that exceed a comfortable threshold, alerting stakeholders of lag or application response issues. Real user monitoring enables an organization to efficiently respond to faults and understand their effect. There are two ways of tracking end user experience:

Synthetic monitoring. This tracking method uses probes and bots to simulate an end user to determine problems before the app is opened. [Synthetic monitoring](https://www.techtarget.com/searchsoftwarequality/definition/synthetic-monitoring) is also used to monitor service-level agreements (SLAs) associated with the app.

Agentless monitoring. This method uses data probes to analyze [network traffic that travels through load balancers](https://www.techtarget.com/searchnetworking/answer/Application-vs-network-load-balancing-Whats-the-difference) and switches. Agentless monitoring reveals information about performance throughout the entire infrastructure, as well as details on the analyzed client -- such as their location, OS and browser.

Business transactions. Also known as user-defined transaction profiling, this component focuses on examining specific user interactions and recreating them to test and understand the conditions that lead to a performance problem. This process helps organizations trace events as they move across the app's various components, as well as revealing when and where events are occurring -- and whether performance efficiency is optimized. This kind of profiling, along with component monitoring, is vital for effective troubleshooting in complex application environments.

Component monitoring. Also known as application component deep dive, this aspect involves tracking all components of the IT infrastructure. Extensive, in-depth monitoring is performed on all the used resources and experienced events within the app performance infrastructure. This includes an analysis of all servers, operating systems, middleware, application components and network components. Component monitoring provides a deeper understanding of the various elements and pathways identified in the previous processes.

Analytics and reporting. This aspect involves translating the data gathered from the above processes into information that can be used for the following:

defining a performance baseline using historical and current data that sets an expectation for normal app performance;

identifying potential areas of improvement by comparing infrastructure changes to performance changes;

efficiently identifying, locating and resolving performance issues using historical and baseline data; and

predicting and alleviating potential future issues using actionable insights.

The analytics and reporting components are essential to ensuring the organization receives a good return on investment (ROI) from both the application and APM.

**How to choose APM tools**

The first step to identifying the right tools is to identify stakeholders and their critical pains.

In other words:

Who will benefit from APM? Look both within and beyond IT, from management to the end user.

What problems do they need to solve? Look at the rate and types of problems, their severity, and the number of employees involved in troubleshooting them.

This can help uncover blind spots in your existing environment. Added to the service level standards you’re legally required to maintain, you should end up with a list of requirements from which to identify must-have tools.

And we’d argue that a comprehensive APM strategy needs only three:

Business-impact analytics to tie application issues to areas for business improvement

End-user monitoring to identify issues before your customer does

AI capabilities to address potential issues before they even occur

So what do these APM tools look like, and how can they impact IT?

Business-impact analytics

Some people call this “full stack tracing”: [connecting the dots between app performance, user experience, and business outcomes](https://www.appdynamics.com/product/business-iq) by collecting, isolating, and correlating code-level details for transactions that take place in your app. The ideal tool automates this for every transaction and delivers insights in real time.

This goes beyond simply monitoring application performance to find out how that performance impacts business operations. Could it be a performance problem that caused a customer to abandon their shopping cart, for example? With that kind of insight, IT could inform the marketing team’s win-back campaigns and help drive superior customer experiences.

End-user monitoring

This ultimate measure involves [tracking the actual experience of your users](https://www.appdynamics.com/product/end-user-monitoring/). What better way to find performance bottlenecks?

There are two main ways to do so: by simulating and testing synthetic user interactions and by passively monitoring the end-user experience as users interact with your applications in real time. The ideal solution offers both.

With these capabilities, app support teams can optimize every touchpoint in the user’s journey by capturing and fixing errors, crashes, page load details, and other critical metrics before the end user notices a problem. This can launch you miles ahead of [the majority of enterprises that are alerted to anomalies by users](https://www.appdynamics.com/blog/engineering/successfully-deploying-aiops-part-1-deconstructing-mttr/).

AI and [machine learning](https://www.appdynamics.com/product/application-performance-management/application-performance-monitoring/cognition-engine)

AI-driven analytics are a relatively new frontier in APM, and has led to the advent of [AIOps](https://www.appdynamics.com/blog/aiops/what-is-aiops/) platforms. In recent years, they’ve developed to include machine learning, statistical analysis, and pattern recognition to automatically diagnose root cause and predict anomalous behaviors. For ease of use, the best APM tools distill these complicated insights into actionable information in a single pane of glass.

New though it may be, AI is indispensable in environments where apps are elastic, highly virtualized, and built with microservices. It can (and should) be a key differentiator in your search. By predicting events, you can squash possible issues before they become issues. Capacity planning becomes much less daunting.

