

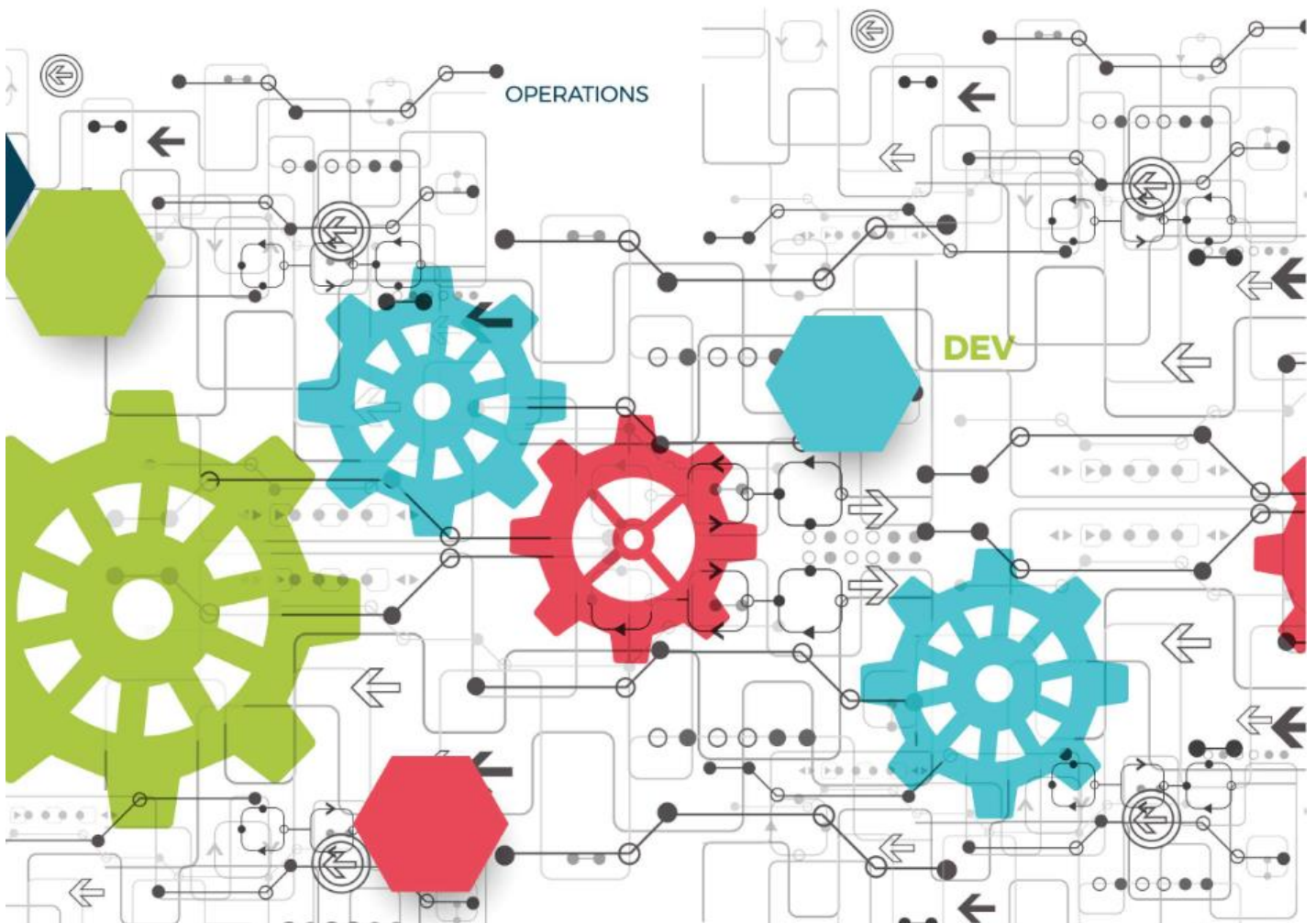


**B.Tech Computer Science
and Engineering in DevOps**

System Provisioning and Configuration Management

MODULE 4

System Provisioning and Configuration Management



Contents

Module Objectives	1
Module Topics	1
4.1 State of Various Tools in Provisioning and Configuration	2
4.2 Infrastructure as Code	2
4.2.1 Importance of Infrastructure as Code	3
4.2.2 Advantages of Infrastructure as Code	4
4.2.3 Operational Procedure of Infrastructure as Code	5
4.2.4 Best Practices of Infrastructure as Code	5
4.3 Continuous Integration/Continuous Deployment	6
4.3.1 Steps to Implement Continuous Integration	7
4.3.2 Benefits of Continuous Integration	8
4.3.3 Steps in Continuous Deployment	8
4.3.4 Benefits of Continuous Deployment	9
4.4 Configuration Management	9
4.5 Configuration Management	10
4.5.1 Why Configuration Management is Important?	10
4.5.2 Benefits of Configuration Management	11
4.6 Configuration Management in DevOps	12
4.6.1 Drawbacks of Configuration Management	12
4.7 Monitoring	13
4.8 What is Monitoring?	13
4.8.1 How Monitoring Supports DevOps?	14
4.9 Reasons for Using Provisioning and Configuration Tools	15
4.10 Automation, Preventing Errors and Tracking of Changes	15
4.10.1 Automation in DevOps	16
4.10.2 Preventing Errors in DevOps	17
4.10.3 Tracking of Changes	18
4.11 Tools and their Capabilities	18
What did you Grasp?	23
In a Nutshell	24



Facilitator Notes:

Welcome the participants and give them an overview of the module. Tell them that they will learn about 'System Provisioning and Configuration Management'.

You will learn about 'System Provisioning and Configuration Management' in this module.

Module Objectives

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

- Explain the state of various tools in provisioning and configuration
- Describe the reasons for using provisioning and configuration tools
- Define automation, preventing errors and tracking of changes
- Discuss examples of tools and their capabilities



Facilitator Notes:

Explain the module objectives to the participants.

Module Topics

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

- State of Various Tools in Provisioning and Configuration
- Reasons for Using Provisioning and Configuration Tools
- Automation, Preventing Errors and Tracking of Changes
- Examples of Tools and their Capabilities



Facilitator Notes:

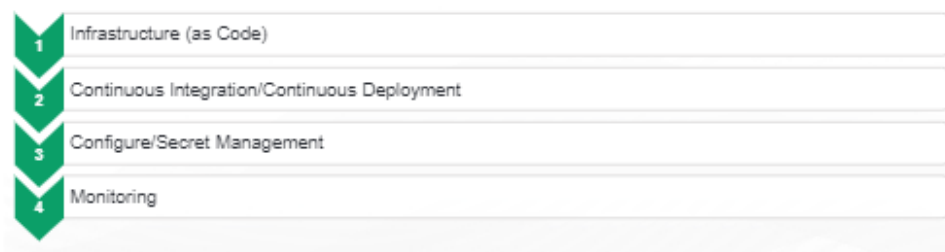
Inform the participants about the topics that they will be learning in this module.

You will learn about the following topics in this module:

- State of Various Tools in Provisioning and Configuration
- Reasons for Using Provisioning and Configuration Tools
- Automation, Preventing Errors and Tracking of Changes
- Examples of Tools and their Capabilities

4.1 State of Various Tools in Provisioning and Configuration

The state of these tools is mostly divided into these categories:



Facilitator Notes:

Inform the participants about the state of various tools in provisioning and configuration.

We will learn about the state of various tools in infrastructure and configuration and understand the segregation of various tools falling under this category and why segregation is important.

4.2 Infrastructure as Code

According to Wikipedia, "Infrastructure as code is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools."

Facilitator Notes:

Explain the participants about infrastructure as code.

There is a saying in normal life "Building something is difficult, but maintaining/managing what you build is even more difficult". This also applies to IT (Hardware+Software) Infrastructure.

So how do we define Infrastructure as Code – if we go via Wikipedia then Wikipedia says "Infrastructure as code is the process of managing and provisioning computer data centers

through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools."

Simply put – Managing IT(Hardware+Software) infrastructure via the help of configuration files.

4.2.1 Importance of Infrastructure as Code

The importance of infrastructure as code are as follows:

- Decreases cloud deployment inconsistencies
- Increases development productivity
- Decreases costs incurred



Facilitator Notes:

Discuss the importance of infrastructure as code with the participants.

Among the various cloud architectures which of course speeds up infrastructure set up and also handles the problem of high scalability, but these are optimum in the case of simpler architectures. In case of complex ones, mistakes do creep in. And the most common mistake is the human error. Because multiple people work on doing the configuration setup and deployments and in such a scenario mistakes are inevitable. To avoid this, generally the best practice is via automation and infrastructure as code. So environments into cloud is set up and launched with minimal mistakes. As errors are quantified in advance in automation test runs.

Infrastructure as the code has played a major role in defining optimal ways where in IT companies manage the IT build, scale and products related to technology. Lesser involvement of multiple people has eradicated discrepancies as clicks over buttons have reduced. Provisioning different servers have become easier. In fact, all these are applicable to other databases and interlinked infrastructures and this has led to an increase in development productivity on a whole.

Reduction in manpower, reduction in managing those manpower, reducing complexities, as well as lot of hardware components because of introduction of infrastructure as code has led to saving a lot of money.

4.2.2 Advantages of Infrastructure as Code

Advantages of infrastructure as code are as follows:



Facilitator Notes:

Explain advantages of infrastructure code to the participants.

Speed: Just by running a script, infrastructure as code helps us to setup our infrastructure very fast (faster than the primitive methods). For every stage of the lifecycle of a product, infrastructure as code helps speed up activities.

Consistency: Human errors are inevitable in the manual traditional mode of things. And in case in the first iteration of setup everything works fine there is no consistent guarantee that it will work in the subsequent iterations as well. But, infrastructure as code uses the same successful configuration files with minimal changes over and over again. So the consistency is higher.

Accountability: You can trace any failure in the configuration file and run it. In the next iteration, you know what changes were implemented. Double check if you need more assurance and run it. So there is always better accountability.

Efficient: The above advantages indirectly proves how efficient infrastructure as code would be. And not only in development, but also in QA and maintenance perspective.

Cost Effective: Less manpower, management also decreases, re-work decreases significantly, maintenance is better, too much crash implications do not exist. Early detection of deviation is there. So all these accounts to lesser money spend.

through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools."

Simply put – Managing IT(Hardware+Software) infrastructure via the help of configuration files.

4.2.1 Importance of Infrastructure as Code

The importance of infrastructure as code are as follows:

- Decreases cloud deployment inconsistencies
- Increases development productivity
- Decreases costs incurred



Facilitator Notes:

Discuss the importance of infrastructure as code with the participants.

Among the various cloud architectures which of course speeds up infrastructure set up and also handles the problem of high scalability, but these are optimum in the case of simpler architectures. In case of complex ones, mistakes do creep in. And the most common mistake is the human error. Because multiple people work on doing the configuration setup and deployments and in such a scenario mistakes are inevitable. To avoid this, generally the best practice is via automation and infrastructure as code. So environments into cloud is set up and launched with minimal mistakes. As errors are quantified in advance in automation test runs.

Infrastructure as the code has played a major role in defining optimal ways where in IT companies manage the IT build, scale and products related to technology. Lesser involvement of multiple people has eradicated discrepancies as clicks over buttons have reduced. Provisioning different servers have become easier. In fact, all these are applicable to other databases and interlinked infrastructures and this has led to an increase in development productivity on a whole.

Reduction in manpower, reduction in managing those manpower, reducing complexities, as well as lot of hardware components because of introduction of infrastructure as code has led to saving a lot of money.

4.2.2 Advantages of Infrastructure as Code

Advantages of infrastructure as code are as follows:



Facilitator Notes:

Explain advantages of infrastructure code to the participants.

Speed: Just by running a script, infrastructure as code helps us to setup our infrastructure very fast (faster than the primitive methods). For every stage of the lifecycle of a product, infrastructure as code helps speed up activities.

Consistency: Human errors are inevitable in the manual traditional mode of things. And in case in the first iteration of setup everything works fine there is no consistent guarantee that it will work in the subsequent iterations as well. But, infrastructure as code uses the same successful configuration files with minimal changes over and over again. So the consistency is higher.

Accountability: You can trace any failure in the configuration file and run it. In the next iteration, you know what changes were implemented. Double check if you need more assurance and run it. So there is always better accountability.

Efficient: The above advantages indirectly proves how efficient infrastructure as code would be. And not only in development, but also in QA and maintenance perspective.

Cost Effective: Less manpower, management also decreases, re-work decreases significantly, maintenance is better, too much crash implications do not exist. Early detection of deviation is there. So all these accounts to lesser money spend.

4.2.3 Operational Procedure of Infrastructure as Code

The operational procedure of infrastructure as code has two approaches:

1. Imperative Approach
2. Declarative Approach



Facilitator Notes:

Inform the participants about operational procedure of infrastructure as code.

Imperative Approach: It kind of gives instructions or orders. There are certain set of commands in a sequential manner. And these commands are carried out in the mentioned sequence to attain the final goal set.

Declarative Approach: This is more like a final result announcement. Sometimes apprehensive, but it mostly points to the desired outcome. A typical explicitly sequential set of commands is not there. But, it more likely shows this is how the end result may look like. It is comprehensive in that fashion.

4.2.4 Best Practices of Infrastructure as Code

Best practices of infrastructure as code are as follows:

- Configuration files should have genuine code
- Version control should be used
- Less documentation
- Validation and monitoring configurations



Facilitator Notes:

Inform the participants about the best practices of infrastructure as code.

Configuration files should have genuine code: The configuration files should be explicitly coded with code which runs as per the specifications mentioned in the infrastructure. They should have genuine and truthful code. This is highly important as the same configuration files will not limit only to one iteration. It will be reused a lot.

Version control should be used: In today's scenario, there is no copy pasting of code and sending in zipped files. Version control is inevitable.

Less documentation: Maintain the ingenuity of the configuration files. That will take care of the infrastructure. Monitor, track and trace the same. Hence, less need of documentation.

Validation and monitoring configurations: Verify your code and make it testable. Implement certain monitoring tools. Check for inconsistencies and deviations. Implement necessary changes and then deploy.

4.3 Continuous Integration/Continuous Deployment

What is Continuous Integration (CI)

When many developers and teams (multiple) are working on different branches of the same web application, they need to perform integration tests. To do that an automated process for each piece of code is performed on a daily basis so that all code gets tested. And this whole process is termed as continuous integration.

What is Continuous Deployment (CD)

It is an extension of Continuous Integration which actually makes some features ready to use (at an Alpha or Beta Level) to some end users. During this process, it passes through several stages of QA, Staging, etc., before it enters into production.

Facilitator Notes:

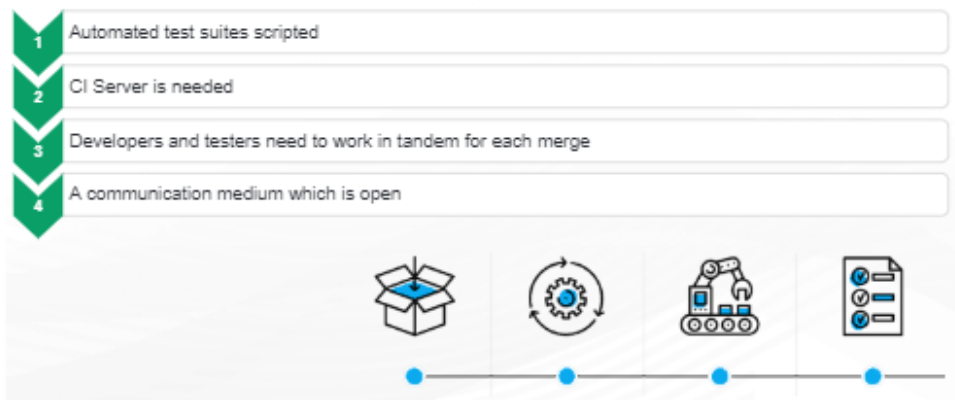
Explain the participants about continuous integration and deployment.

Continuous Integration: It is a development practice where a developer or group of developers integrate the code into a repository (GitHub Or SVN) into a certain branch. Mostly they integrate into Master Branch or Main Branch. And the frequency of integrating the code into the repository depends on the number of changes implemented, the number of defects fixed, number of deviations traced, etc.

Continuous Deployment: It is a practice wherein the customer gets to see the shippable product. In software sense, the release is released from the UAT to production in the pipeline and all the change requests, stories, old defects have been taken care of, and it is released to the customer. It is a deployment which is more of a click of a button that triggers the process and there is no human intervention. If failed, then it is fixed and deployed again.

4.3.1 Steps to Implement Continuous Integration

The steps to implement continuous integration are as follows:



Facilitator Notes:

Inform the participants about the steps to implement continuous integration.

For successful implementation of continuous integration, testing team has a certain responsibility too. Although, agreed on paper and on the definition, QA team comes later, but even though testing team's participation is strictly not a part of CI, but it is anyways implied. So the testing team has to write automated scripts for every new feature or Change Request (CR). In case there is a bug fix, and some automation scripting needs to change even if it is a small validation change, then that change has to be implemented.

A continuous integration server is needed, which can have a vigil on the main repository and should be connected with the repository so that it runs the tests automatically for every new commits pushed. For e.g., JENKINS has to be linked with GitHub and the moment a code is pushed into GitHub then JENKINS runs its respective jobs.

Developers and testers need to push their changes as soon as they finish the code fix, or defect fix or any CR. The frequency varies from project to project, but the changes have to be done and integrated.

There is a shout out communication medium where the developers and testers who are done with their code changes and are about to integrate them into the respective branches notify the team in a communication medium like group chat to announce what all changes are getting configured.

4.3.2 Benefits of Continuous Integration

Following are the benefits of continuous integration:



Facilitator Notes:

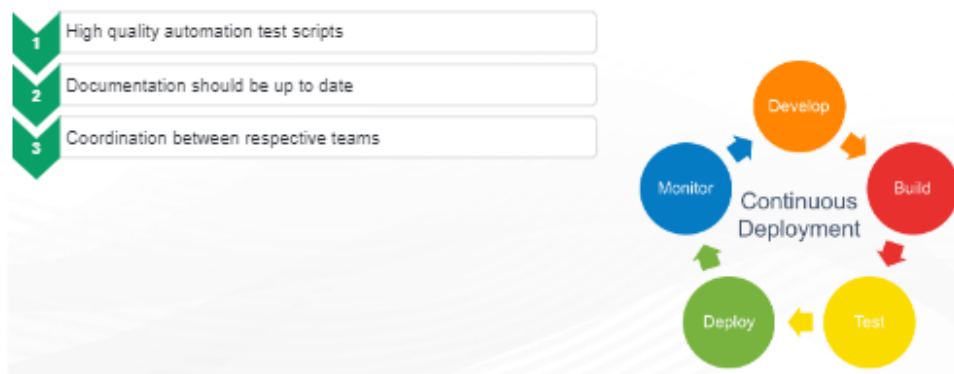
Inform the participants about the benefits of continuous integration.

Fewer defects get shipped to production as defects/bugs/deviations are identified at the regression level in the automated tests. Building the release gets easier as integration issues are resolved at an early stage. Context switching is much less as developers get an alert in case a build breaks and they start fixing it as soon as possible.

QA costs are reduced exponentially as the CI server can run a lot of tests in the matter of minutes.

4.3.3 Steps in Continuous Deployment

Steps to implement continuous deployment are as follows:



Facilitator Notes:

Explain the steps in continuous deployment to the participants.

QA Team has to play an active part here. They need to be on their toes and be at their best. The quality of test suite will determine the quality of releases. Documentation process needs to keep up with the pace of deployments. Even if it is in the form of a test report sheet or test summary sheet. Feature flags are possible and teams should be prepared for both upside and downside. So a clarified coordination is highly important.

4.3.4 Benefits of Continuous Deployment

Following are the benefits of continuous deployment:



Facilitator Notes:

Inform the participants about the benefits of continuous deployment.

There is no much pause and play of development for releases. Deployment pipelines are triggered automatically for every change which is made in the repository branch. Releases are less risky and easier to fix in case of problems as you deploy small batches of changes. If the change is substantial then developers can push the code by Version Control Tool, for example, Git into the repository, i.e., GitHub and keep a track of the changes implemented. This push is directly synced with the CI tool, for example, JENKINS and the moment it acknowledges the change pushed into GitHub, then JENKINS runs its job or jobs as per the Architecture and the deployment begins in an automated way.

Imagine the time saving ability and also the reduction in costs for handling such a long operation which is done in one go via continuous deployment methods.

4.4 Configuration Management

Topics to be covered in this section are as follows:

- 1 What is Configuration Management?
- 2 Why is it Important?
- 3 Benefits of Configuration Management
- 4 Configuration Management in DevOps
- 5 Drawbacks of Configuration Management



Facilitator Notes:

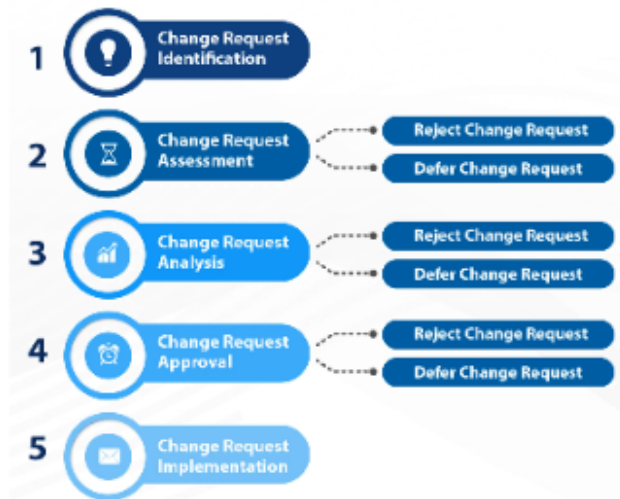
Inform the participants about the topics that they will be learning in this section.

In the upcoming slides, we will read about the configuration management process for a product as an end-to end perspective.

4.5 Configuration Management

What is Configuration Management?

- In IT Software, configuration management is a unique task. It is done for deployment, configuration and maintenance of servers.
- It is a system engineering process. It provides visibility and control of a system's performance, functional, and physical attributes.



Facilitator Notes:

Explain about configuration management to the participants.

It is a system engineering process. In retrospection of a product's attributes such as requirements, design, build, test, integration, deploy — configuration management tries to establish and maintain consistency in the product's functional behavior and performance output.

4.5.1 Why Configuration Management is Important?

Various configuration management tools help us achieve the below benchmarks.

When we talk about multiple software builds, releases, revisions, and versions, then we ought to explain the need for storing and maintaining the data, keeping track of the development builds and simplified troubleshooting.



Facilitator Notes:

Explain the participants why configuration management is important.

Here, we will learn why configuration management is important.

- In software, configuration management is called Software Configuration Management (SCM) or Unified Configuration Management (UCM).
- SCM is important because it helps developers to keep track of source code, implemented code, documents and artifacts, enhancements, change requests and changes incurred.
- SCM also helps in giving a structural approach to development activities such as status reporting on development processes.
- SCM acts as a bridge between physical servers and operating system instances.

4.5.2 Benefits of Configuration Management

Following are the benefits of configuration management are as follows:

- SCM helps in understanding impact change in any environment by doing a change impact analysis
- SCM helps in restoration of services at a faster rate
- SCM helps in optimizing the impact time during failures
- SCM helps in auditing and compliance support
- SCM helps in Cost optimization

**Facilitator Notes:**

Discuss the benefits configuration management to the participants.

- SCM helps in understanding impact change in any environment by doing a change impact analysis
- SCM helps in restoration of services at a faster rate
- SCM helps in optimizing the impact time during failures
- SCM helps in auditing and compliance support
- SCM helps in Cost optimization

4.6 Configuration Management in DevOps

Now, let's learn about configuration management in DevOps.



Facilitator Notes:

Inform the participants about configuration management in DevOps.

Identification: Identification is the process that caters to the need of identifying the system's requirements. Such as what configurations, it actually needs. What is the optimum configuration set up and then it catalogs the same.

Control: One of the purpose of configuration management is to bring about control in the overly changing configurations in the methodologies which predated DevOps. In the DevOps scheme of things, it is highly unlikely that the change in configurations will get misaligned or destabilize. This is because configuration control allows the changes to happen in a controlled manner, maintaining the stability of the integrations of the existing infrastructure.

Audit: it is like other audit processes. Has to follow the norms and regulations of various compliances and other industry standards.

4.6.1 Drawbacks of Configuration Management

Following are the drawbacks of configuration management:

- 1 Multiple tools are necessary
- 2 Skilled technicians or engineers
- 3 Initial stages of architecture are challenging
- 4 Implementing change might be an uphill task



Facilitator Notes:

Explain the participants about the drawbacks of configuration management.

Configuration management is not an easy proposition. It might sound easy and formatted and structure on paper, but in the real project world scenario it is a complicated thing. Imagine the number of standards and procedures that needs to be followed in the architectural flow. The timing of implementation and execution needs both skilled people and knowledge on automation tools to get it done. The investment should be on skill enhancement of people. Unskilled people cannot get the job done.

If in the beginning state of architecture flaws creep in then the whole process goes for a toss. It is imperative to keep a balance on all the stages of the pipeline. It has to be pin point accurate which is quite a challenge.

And even though it speeds up everything, a new change might take some doing to implement and get stability and see the old functionality workaround has not been affected.

4.7 Monitoring

The topics to be covered in this section are as follows:



Facilitator Notes:

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

In the upcoming slides, we will understand the role of monitoring in DevOps pipeline and answer the question "What do we Monitor"?

4.8 What is Monitoring?

With a cycle of CI/CD, DevOps association with a discharge cycle tends to get shorter. Each change is reflected into the general conditions quite promptly and these are done with a keen observation to get appropriate client feedback. So the idea of constant checking has been utilized to assess every application execution progressively. This is called continuous monitoring. The idea of constant verification to asses every action of an application which has a progressive lifecycle.

Facilitator Notes:

Inform the participants about monitoring.

Monitoring is a process which predates DevOps. But, off late we have moved on from writing application code to the infrastructure as code, automation, integration, testing, deployment everything is happening. Technically, the speed has become quite fast and this results in overloading on customer feedbacks and also on various deployment tools. So technically speaking, we need to monitor not only automation test scripts, integration, testing, provisioning, deployment, but also our builds, resources and also performance.

So monitoring in DevOps can be simply called as — “Vigil on all the branches which means every stage of DevOps pipeline needs monitoring and vigilance and made more secure”.

4.8.1 How Monitoring Supports DevOps?

Now, let's learn how monitoring supports DevOps.

**Facilitator Notes:**

Explain the participants how monitoring supports DevOps.

Collaborate: Monitoring earlier in the DevOps lifecycle helps teams to collaboratively work to improve performance, reduce scalability and optimize availability. Can use artifacts from the design phase and test phase to create certain monitors that can be used for vigilance in pre-production, production environments and in some cases maintenance in post-production. Teams can come together to standardize stuff and get a better product.

Automate: Proactive monitoring is an automation process in itself. If applications are up and running as expected, conclusive statement is — monitoring tool itself is functioning properly. From a team's perspective, the earlier the monitoring the better it is. And manual is an extended time taken and human intervention processes which can lead to more human errors and discrepancies. So more the automation the better.

Configure: A proactive monitoring tool provides configurability measures. This is mostly for monitor creations and also to set up alerts. A monitoring tool helps make the process more tailored and error free. It helps in making it more efficient — whether it is from a specified

location or from an organized work group.

Share: Monitoring helps in providing teams with a common data set. This data set can be shared across various stages of the application with the prime intention to improve it.

4.9 Reasons for Using Provisioning and Configuration Tools

Following are the reasons for using provisioning and configuration tools:



Facilitator Notes:

Explain the reasons for using provisioning and configuration tools to the participants.

Following are the reasons for using provisioning and configuration tools:

- Achieve faster application delivery
- Decrease scalability
- Manage infrastructure as code
- Decrease human intervention
- Increases efficiency
- Traceability is better
- Reusable with minimal edits
- Cost effective

4.10 Automation, Preventing Errors and Tracking of Changes

Here, we will learn about automation, preventing errors and tracking of changes.



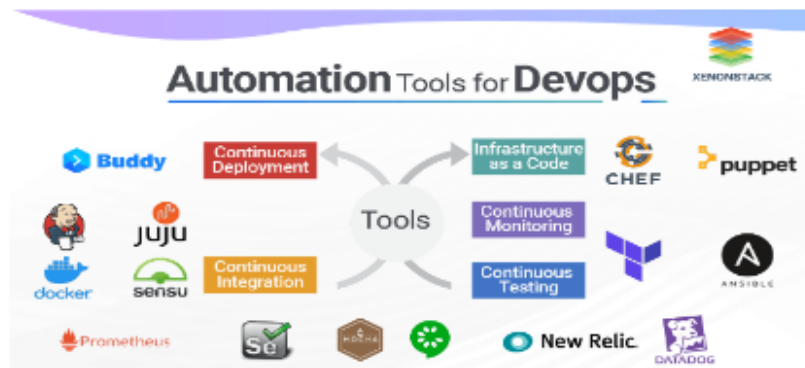
Facilitator Notes:

Inform the participants about automation, preventing errors and tracking of changes.

In the upcoming slides, we will learn how the DevOps tools play a pivotal role in automating, error detection and prevention and also keep a track of code changes in configuration setup files.

4.10.1 Automation in DevOps

In DevOps automation begins from code generation in the developer's machine. It continues till the code is in production and even during the monitoring phase even after the application has gone live.



Facilitator Notes:

Inform the participants about automation in DevOps.

Role of automation in DevOps is very important not only in the perspective of process, but also delivering value to the customer at regular intervals. The situation pre-dating DevOps had lots of expert engineers working on network connectivity, setup and configuration of servers and deployment of the same. And most of it was manual work with a lot of manual workarounds, and pre-defined set up rules which cannot be tampered unless advised and again those advisories included manual intervention.

So even though the bright minds were working on it, the brightest minds were manually doing it and with manual work, there is no guarantee of great efficiency and error free delivery. One mistake leads to another. Thus, automation had to be introduced.

In DevOps automation begins from code generation in the developer's machine. It continues till the code is in production and even during the monitoring phase even after the application has gone live.

Automation in terms of testing is a critical aspect. Unit tests by developers, installation tests, integration tests, user acceptance tests, UI tests everything is automated.

Tools which help in automation testing — few to name which are really popular in the market are Selenium in case of browser tests, Appium in case of mobile tests.

4.10.2 Preventing Errors in DevOps

The errors which needs prevention is broadly classified as follows:



Facilitator Notes:

Describe the participants about preventing errors in DevOps.

Error prevention in DevOps is (not only) related to fixing a malice code or fixing an error and trying to optimize it further so that we need not encounter the same error again. This is a very tiny approach towards error prevention in DevOps.

The main areas of concern are the following:

1. **Oversimplification of DevOps process:** DevOps is a transition. It cannot be an overnight overhaul and change it to what you were using. DevOps needs training and education, rather than hiring DevOps engineers and expect them to develop a process and make things work. The best people are already the ones inside the system who know the product, its behaviour and expectations of the client. All what organizations need to do is train them into the DevOps scheme of things. Invest the money in training rather than hiring fresh candidates and expect them to change the scenario.
2. **Rigidity:** This kind of acts like Achilles heel in some DevOps processes. Organizations cannot be rigid with something which has worked in the past, there is no assurance that will work in the present. They have to make proactive adjustments, especially catering to industry demands.
3. **Indecisiveness in using tools:** Focusing on too many tools, to do too many different jobs does not make sense. It only complicates matter. Rather than using many teams should focus on what tool exactly you want to use.
4. **Focusing on speed rather than quality:** Obviously DevOps is about speed and saving time, but increasing speed at the expense of quality and security is a big mistake. So testing should be done at an earlier stage. On top of that release build should be ready for continuous delivery before deploying.

There are other aspects to error prevention, but in case the architecture design goes wrong, then from the beginning everything is haywire. So that becomes a catastrophe not error.

4.10.3 Tracking of Changes

The changes that should have a constant monitoring and a tracking mechanism are as follows:



Facilitator Notes:

Inform the participants about tracking of changes.

Just like error prevention should be undertaken at every stage of the DevOps Pipeline, similarly tracking of changes should also be taken care of at every stage. Whenever the code is generated from the Developer' machine, then also the track of the running code is kept in tact. In case there is a new CR and code has to be changed and for some reason the new code does not work, then the old code base could be revived. All this is possible using automation tools such as Git, etc.

Tracking of changes should happen in each and every stage. Whether it is development, unit tests, implementation tests, integrations tests, UATs or even going into production and post production monitoring via monitoring tools on the live product. Tracking has to be present at every stage.

4.11 Tools and their Capabilities

Here, we will discuss about tools and their capabilities.

- Provisioning, Version Control, Configuration Management Tools
- Software Testing Tools
- Deployment Tools
- Release Management Tools
- Monitoring Tools
- Log Management Tools
- Process supervision Tools
- Feedback Tools
- Container Tools
- Cloud Computing Tools
- Security Tools
- Data and Business Intelligence Tools



Facilitator Notes:

Inform the participants about tools and their capabilities.

There are various tools and technologies that enables companies to optimize DevOps practices as a Complete End-to-End Methodology. DevOps promotes speed, automation, less human intervention, efficiency, reliability, traceability, ease of maintenance and cost optimization. So the right set of tools and technology makes DevOps what it is.

Provisioning, Version Control, Configuration Management Tools

Ansible	Consul	ISPW	Rudder
AWS CloudFormation	Crucible	Kallithea	Rundeck
Bofg2	Fisheye	Mercurial	Salt
BladeLogic	Foreman	Packer	Stash
CFEngine	Gerrit	PalletOps	Subversion
Chef	Git	Perforce	Terraform
Cobbler	GitLab	Puppet	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the provisioning, version control and configuration management tools which are available in the market in certain areas of DevOps.

Software Testing Tools

Appium	HP UFT	Mocha	Selenium
Bosun	IBM Rational Quality Manager	Parasoft Environment Manager	SonarQube
Codacy	Jasmine	Parasoft SOAtest	SpecFlow
Cucumber	JMeter	Protractor	TestComplete
FitNesse	JUnit	Pytest	TestNG
Gatling	Karma	QUnit	Tosca
Gauntlit	Locust	Sahi	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the software testing tools which are available in the market in certain areas of DevOps. Some of the tools that are on top of the market are as follows:

1. Appium
2. Selenium
3. TestNG
4. Junit
5. TestComplete

Deployment Tools

Automic	Deploybot	Iulu	RapidDeploy
AWS CodeDeploy	Deployer	Nomad	Rundeck
BMC Release	ElectricFlow	Octopus Deploy	SmartFrog
CA Nolio	Go	Otto	XL Deploy
Containership	Google Deployment Manager	Rancher	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the development tools which are available in the market in certain areas of DevOps. Capistrano is on top of the market.

Release Management Tools

BMC Release Process Management	Datical	Plutora	Xebialabs
CA Nolio	DBmaestro	Serena	XL Release
CA Release Automation	LaunchDarkly	UrbanCode Release	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the release management tools which are available in the market in certain areas of DevOps.

Monitoring Tools

AppDynamics	Elasticsearch	Kibana	SPM
BigPanda	Ganglia	Nagios	StackState
Blue Medora	Grafana	PagerDuty	VictorOps
Bosun	Graphite	Pingdom	xDatters
Cacti	Hipchat	Riemann	Zabbix
CloudMunch	HostedGraphite	Sensu	
Datadog	Icinga	Sentry	
Dynatrace	jKool	Splunk	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the monitoring tools which are available in the market in certain areas of DevOps.

Log Management Tools

Graylog	Logsense	Sentry	vRealize Log Insight
Logentries	Logstash	Stackify	
Loggly	Papertrail	Sumo Logic	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the log management tools which are available in the market in certain areas of DevOps.

Process supervision Tools

Blue Pill	Monit	Supervisor	Upstart
God	runit	systemd	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the process supervision tools which are available in the market in certain areas of DevOps.

Feedback Tools

Flowdock	Hootsuite	ServiceNow	SurveyMonkey
GetFeedback	Iira Service Desk	Slack	Team Foundation Server
Hipchat	PivotalTracker	StackStorm	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the feedback tools which are available in the market in certain areas of DevOps.

Container Tools

Apache Mesos	Kubernetes	Packer	Swarm
CloudSlang	Linux Containers (LXC)	Rancher	Tectonic
Containership	Nomad	rkt ("rocker")	
Docker	OpenVZ	Solaris Containers	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the container tools which are available in the market in certain areas of DevOps.

Cloud Computing Tools

Amazon Web Services (AWS)	fabric	Microsoft Azure	Pivotal Cloud Foundry
CenturyLink ElasticBox	Google Cloud	OpenStack	Rackspace

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the cloud computing tools which are available in the market in certain areas of DevOps. AWS is the most in demand.

Security Tools

Arachni	Fortify	OWASP ZAP	Sonatype
Brakeman	Gauntlit	Red October	Sqreen
Code Climate	Gemnasium	SecureAssist	Tripwire
credstash	Keywhiz	sneaker	Trousseau
CyberArk	Metasploit	Snorby Threat Stack	Vault
Evident.io	Nmap	Snort	Veracode

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the security tools which are available in the market in certain areas of DevOps.

Data and Business Intelligence Tools


Datadog	Delphix	jKool	Redgate
Datical	Flyway	Kibana	Tableau
DBmaestro	Idera	Liquibase	

Facilitator Notes:

Inform the participants about tools and their capabilities.

These are the data and business intelligence tools which are available in the market in certain areas of DevOps.


What did you Grasp?



- State True or False
Ansible is not a Configuration Management Tool?
A. True
B. False

Facilitator Notes:

Answer: B. False



2. Configuration Management helps us in

- (i) Deploying Servers
- (ii) Configuring Servers
- (iii) Maintaining Servers

A. (i) is true
B. (i) and (ii) are true but (iii) is false
C. Only (iii) is true
D. All of the above

Facilitator Notes:

Answer: B. All of the above

In a Nutshell

In this module, you have learned:

1. State of Various Tools in Provisioning and Configuration
2. Reasons for Using Provisioning and Configuration Tools
3. Automation, Preventing Errors and Tracking of Changes
4. Examples of Tools and their Capabilities



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.

