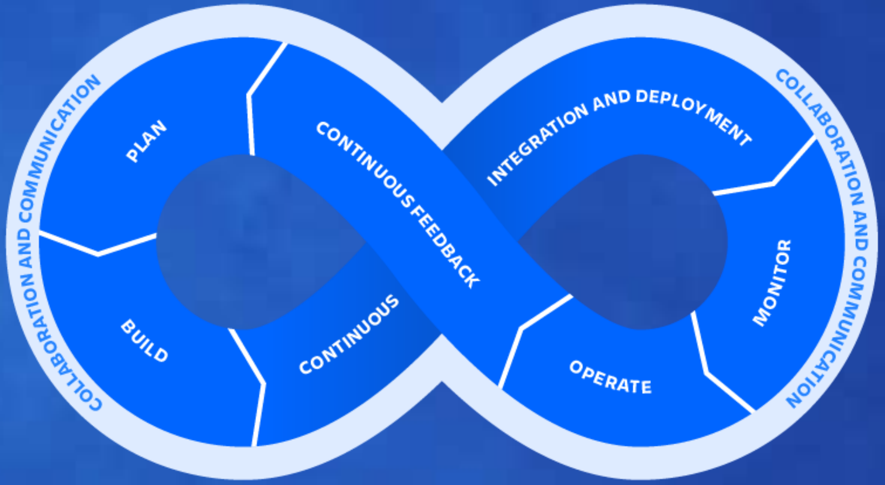

COLLABORATION AND COMMUNICATION
PLAN
BUILD
CONTINUOUS FEEDBACK
CONTINUOUS
INTEGRATION AND DEPLOYMENT
OPERATE
MONITOR
COLLABORATION AND COMMUNICATION

Session 2. Overview of DevOps

Ram N Sangwan

Agenda

- Introduction and Benefits of DevOps
- DevOps Terminologies
- DevOps Life Cycle
- DevOps Concepts and Practices
- Agile Methodology
- Test-Driven Development
- DevOps Tools and Environments

What Is DevOps?

- A practice that allows a single team to manage the entire application development life cycle,
  - Developments, tests, deployments and operations.
- The aim of DevOps is to shorten the system’s development life cycle while delivering features, fixes and updates frequently, in close alignment with business objectives.

DevOps Benefits

Improved Customer Experiences

- DevOps produce faster results in development cycles, controls the costs, and improves ROI.

Digital Transformation

- DevOps is a strategy of successful digital transformation for every industry using technology innovations.

Breaking down silos

- DevOps increases collaboration among team members from all disciplines responsible for product development or delivery and results in more stable business environments.

DevOps Benefits

Easy detection of defects

- With the intervention of DevOps collaboration, code defects are detected and resolved much quicker than your expectations.

Faster Development Cycles

- With DevOps collaboration and automation features, improvements are achieved in the shorter development lifecycles and make sure that products can be released reliably at the right time.

DevOps Terminologies

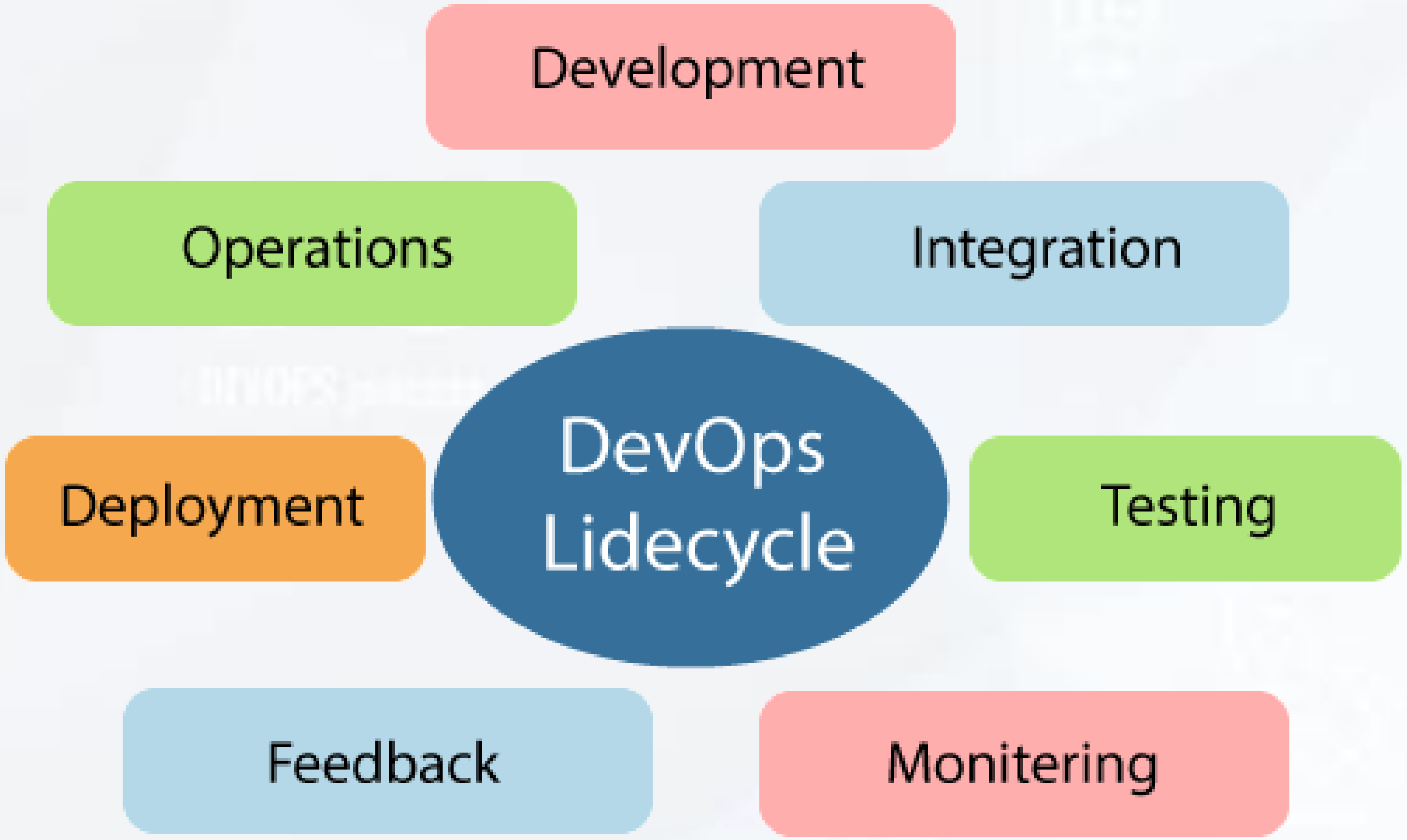
- **Agile**. Used in the DevOps world to describe infrastructure, processes or tools that are adaptable and scalable.
- **Continuous delivery**. A software delivery process wherein updates are planned, implemented and released to end-users on a steady, constant basis.
- **Continuous integration**. A process that allows software changes to be tested and integrated into a code base on a continuous basis each time a change is made to code.
- **Immutable infrastructure**. An application service or hosting environment that, once set up, cannot be changed.
- **Infrastructure-as-Code**. An approach to infrastructure configuration that allows DevOps teams to use scripts in order to provision servers or hosting environments automatically.

DevDevOps Terminologies

- **Microservices**. A type of application architecture in which applications are broken into multiple small pieces.
- **Serverless computing**. A type of service that provides access to computing resources on demand, without requiring users to configure or manage an entire server environment. AWS Lambda, Azure Serverless Functions and Oracle Cloud Functions are some of the examples.

DevOps Lifecycle

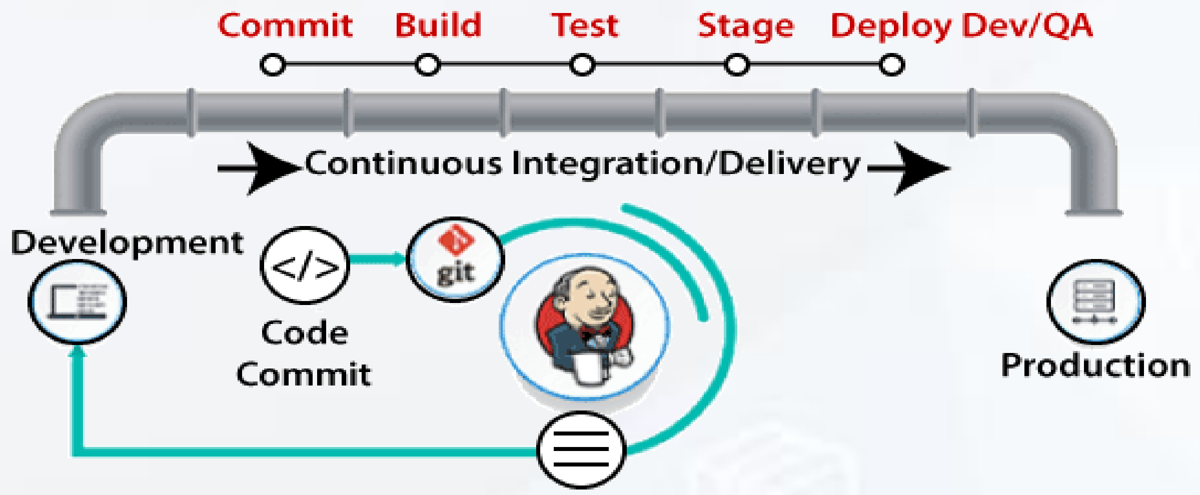
- DevOps defines an agile relationship between operations and Development.



The DevOps lifecycle Phases

1. **Continuous Development** - This phase involves the planning and coding of the software. There are no DevOps tools that are required for planning, but there are several tools for maintaining the code.
2. **Continuous Integration** - The code supporting new functionality is continuously integrated with the existing code.

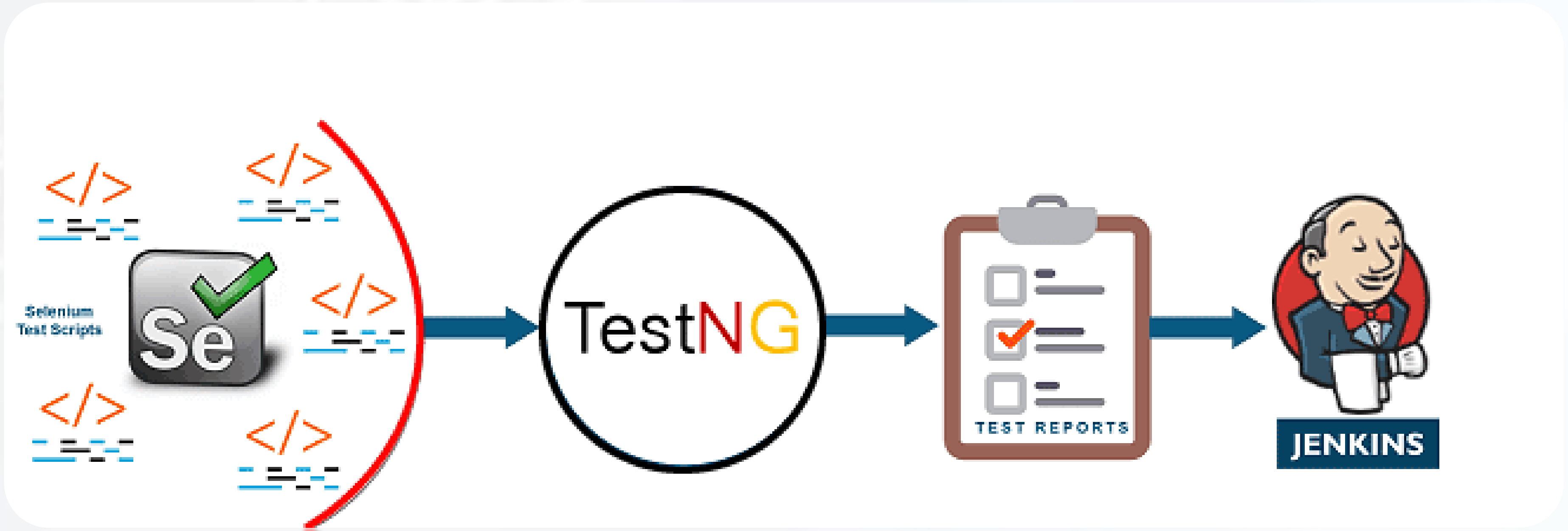
   Jenkins is used in this phase. Whenever there is a change in the Git repository, then Jenkins fetches the updated code and prepares a build of that code, which is an executable file in the form of war or jar. Then this build is forwarded to the test server or the production server.



The DevOps lifecycle Phases

3 **Continuous Testing –** In this phase, the developed software is continuously tested for bugs. Aautomation testing tools such as TestNG, JUnit, Selenium, etc are used in this phase.

Selenium does the automation testing, and TestNG generates the reports. This entire testing phase can automate with the help of a Continuous Integration tool, ***Jenkins***.



The DevOps lifecycle Phases

4. **Continuous Monitoring** – It involves all the operational factors of the entire DevOps process. Important information about the use of the software is recorded and carefully processed to find out trends and identify problem areas.

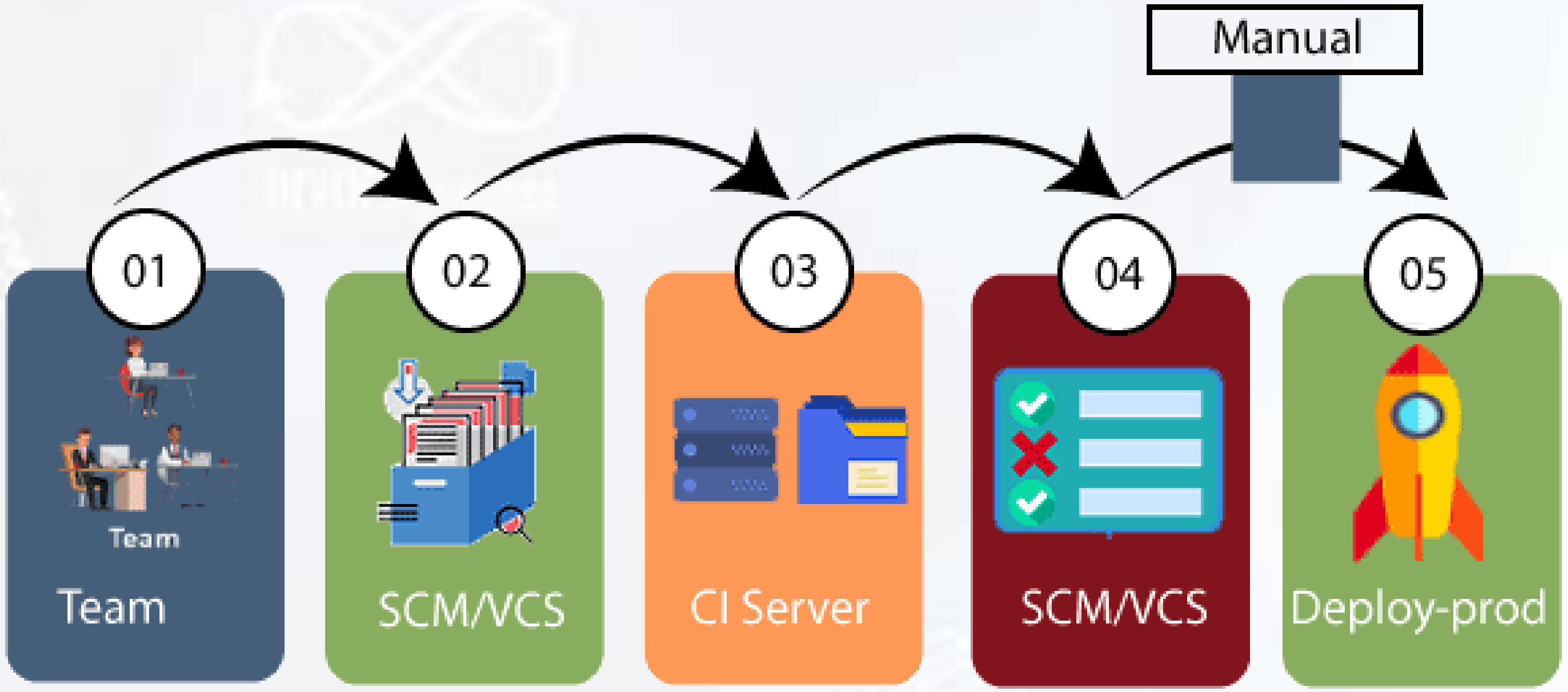
5. **Continuous Feedback** -The application development is consistently improved by analyzing the results from the operations of the software by placing the critical phase of constant feedback between the operations and the development of the next version of the current software application.

The DevOps lifecycle Phases

6. **Continuous Deployment** - The code is deployed to the production servers.

   Some popular tools for this phase are Chef, Puppet, Ansible, and SaltStack.

   Containerization tools are also playing an essential role in the deployment phase. Docker and Podman are popular tools that are used for this purpose.



The DevOps lifecycle Phases

7. **Continuous Operations** - All DevOps operations are based on the continuity with complete automation of the release process and allow the organization to accelerate the overall time to market continuingly.

Agile Methodology

Agile methodology is based on these beliefs

- Current software development processes are too heavyweight or cumbersome
  - Too many things are done that are not directly related to software product being produced
- Current software development is too rigid
  - Difficulty with incomplete or changing requirements
  - Short development cycles (Internet applications)
- More active customer involvement needed

Manifesto for Agile Software Development

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to the value:

1. **Individuals and interactions** over ***processes and tools***
2. **Working software** over ***comprehensive documentation***
3. **Customer collaboration** over ***contract negotiation***
4. **Responding to change** over ***following a plan***

That is, “**while there is value** in the items on the **right**, we value the items on the **left** more.”

Agile Principles

1. *Highest Priority is to Satisfy the Customer through Early and Continuous Delivery of Valuable Software*
2. *Welcome Changing Requirements, even late in Development.*
3. *Deliver Working Software Frequently (From a couple of weeks to a couple of months with a preference to the shorter time scale).*
4. *Business People and Developers Must Work Together Daily throughout the Project.*
5. *Build Projects around Motivated Individuals. (Give them the environment and support they need, and trust them to get the job done.)*
6. *The Most Efficient and Effective Method of Conveying Information to and within a Development Team is face-to-face Communications.*

Agile Principles

7. *Working Software is the Primary Measure of Progress*
8. *Agile Processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9. *Continuous Attention to Technical Excellence and Good Design enhances Agility.*
10. *Simplicity – the art of maximizing the amount of work not done – is essential.*
11. *The Best Architectures, Requirements, and Designs emerge from Self-Organizing Teams*
12. *At regular Intervals, the Team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

Test Driven Development

- TDD or Test Driven Development is a software development approach where a developer writes a test before writing any code.
- The test is then be used to build only the code needed to pass it.
- Once the new code passes the test, it can then be refactored to an acceptable standard as needed.
- TDD ensures that the source code is thoroughly unit tested and helps lead to modularized, flexible and extensible code.
- By focusing on writing only the code necessary to pass the tests, we are inherently making the design simple and clear.
- It’s testing the independent small units or objects to make sure each works as intended.

WHY TDD?

"If it's worth building, it's worth testing. If it's not worth testing, why are you wasting your time working on it?"

- A significant advantage of TDD is that it enables you to take small steps when writing software.
- It is much easier to find, and then fix, defects if you've written two new lines of code than two thousand
- Especially handy in large-scale projects

How OCI Enables DevOps 1/2

- **Oracle Developer Cloud Service for DevOps**
  - Provides a hosted team development and delivery platform including issue tracking, Git code versioning, wiki, Agile development and planning tools, continuous integration and delivery automation with pipelines.
- **Oracle Cloud Infrastructure Resource Manager for IaC**
  - A fully managed service for provisioning infrastructure resources using Terraform.

How OCI Enables DevOps 2/2

- **DevOps Tools and Plugins**
  - Oracle Cloud Platform Terraform provider:
    - To interact with Oracle Cloud Infrastructure resources using Terraform.
  - Terraform modules:
    - Self-contained packages of Terraform configurations, managed together and used to create reusable components.
  - Ansible modules:
    - To provide orchestration, provisioning, and configuration management, using Ansible, for Oracle Cloud Infrastructure services and resources (compute, load balancing, database, and so on.
  - Chef plugin:
    - To provide an interface between Oracle Cloud Infrastructure and Chef server.

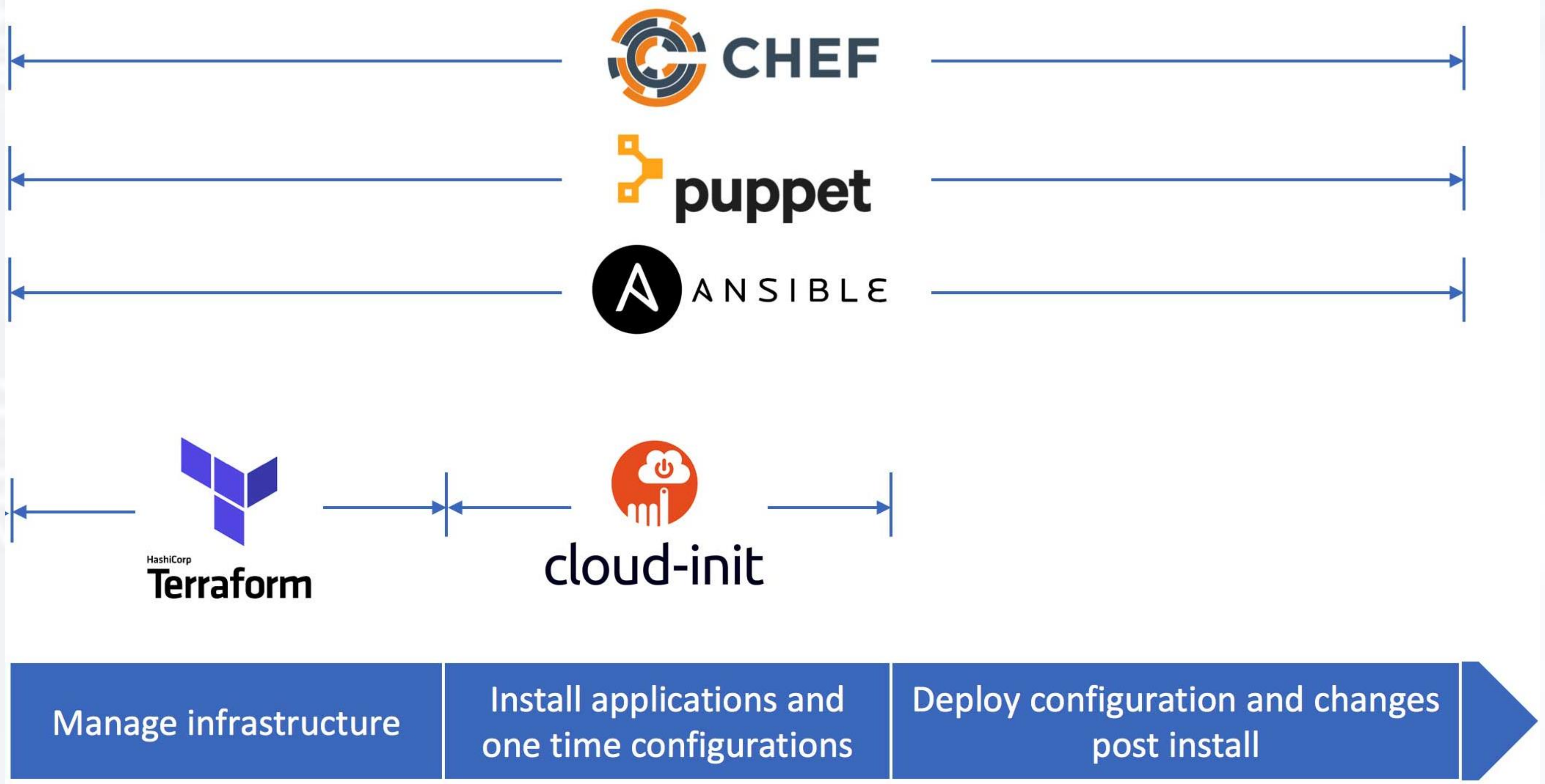
Terraform

- Terraform is an infrastructure provisioning tool created by Hashicorp.
- It allows you to describe your IaC, creates “execution plans” that outline exactly what will happen when you run your code, builds a graph of your resources, and automates changes with minimal human interaction.
- Uses domain-specific language - Hashicorp Configuration Language, HCL.

Ansible

- Ansible is an infrastructure automation tool created by Red Hat.
- Ansible models your infrastructure by describing how your components and system relate to one another, as opposed to managing systems independently.
- Ansible doesn’t use agents.
- Code is written in YAML in the form of Ansible Playbooks, so configurations are very easy to understand and deploy.

A Comparative Study


CHEF
puppet
ANSIBLE
HashiCorp Terraform
cloud-init
Manage infrastructure
Install applications and one time configurations
Deploy configuration and changes post install

Thank You