BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

Review CS#7

Continuous Monitoring

- Introduction to Monitoring
- What to Monitor
- Goals of Monitoring
  - Failure detection
  - Performance degradation
  - Capacity planning
  - User Interaction
  - Intrusion detection
- How to Monitor
- Challenges in Monitoring
- Monitoring Tools
- ELK
- ELK Architecture
- ELK Features and Benefits

Agenda

Configuration Management

- Introduction to Configuration [Infrastructure Concept]
- Managing Application Configuration
- Managing Environments
- Infrastructure as code
- Server Provisioning
- Automating and Managing Server Provisioning
- Configuration management tools- Chef, Puppet
- Managing on-demand infrastructure
- Auto Scaling

Configuration Management

Introduction to Configuration

- Started in the 1950s by the United States Department of Defense as a technical management discipline

  Process of establishing and maintaining the consistency of something's functional and physical attributes as well as performance throughout its lifecycle

- ***Configuration management refers to the process by which all artifacts relevant to your project, and the relationships between them, are stored, retrieved, uniquely identified, and modified***

  This includes the policies, processes, documentation, and tools required to implement this system of consistent performance, functionality, and attributes

Introduction to Configuration Management

Identify your CM strategy

Can I ***exactly reproduce any of my environments***, including the version of the operating system, its patch level, the network configuration, the software stack, the applications deployed into it, and their configuration?

Can I ***easily make an incremental change*** to any of these individual items and ***deploy the change to any, and all, of my environments?***

Can I easily see each change that occurred to a particular environment and trace it back to see exactly what the change was, who made it, and when they made it?

Is it easy for every member of the team to get the information they need, and to make the changes they need to make?

Introduction to Configuration Management

Managing Software Configuration

Configuration information

Used to change the behavior of software at build time, deploy time, and run time

Delivery teams need to consider carefully:

- What configuration options should be available
- How to manage them throughout the application's life,
- How to ensure that configuration is managed consistently across components, applications, and technologies

Treat configuration of your system in the same way you treat your code

Introduction to Configuration Management

Facts and Myth

Flexibility & Configuration

Everyone wants flexible software

Flexibility usually comes at a cost

The desire to achieve flexibility may lead to the common antipattern of “ultimate configurability” which is, all too frequently, stated as a requirement for software projects

Myth

Configuration information is somehow less risky to change than source code

Introduction to Configuration Management

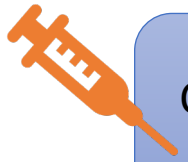
Configuration Vs Source Code

One can stop system easily by changing the configuration compare to by changing the source code

Source Code	Configuration information
Change the source code	Change the Configuration
The compiler will rule out nonsense	configuration information is free-form and untested
And automated tests should catch most other errors	Ex: Change in the URL

Introduction to Configuration Management

Injecting Configuration information



Configuration information can be injected into application at several points

Build Deploy Test Release

Generally, we consider it bad practice to inject configuration information at build or packaging time

It is usually important to be able to configure your application at deployment time so that you can tell it where the services it depends upon such as database, messaging servers, or external systems belong

Manging Configuration

Level of Configurations

- Application Configuration
- Environment Configuration

Manging Application Configuration

How does it works?

Consideration

How do you represent your configuration information?

How do your deployment scripts access it?

How does it vary between environments, applications, and versions of applications?

Considerable Solution

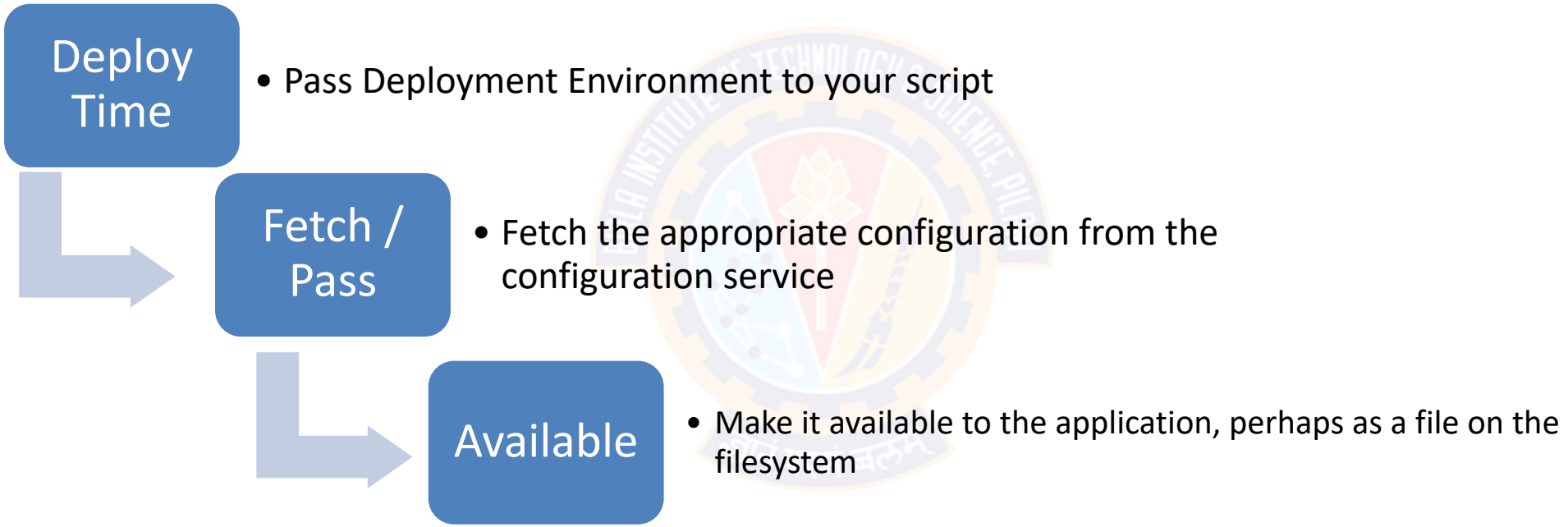
Configuration information is often modeled as a set of name-value strings

- A database
- A version control system
- Or A directory or registry

Note: Version control is the easiest, just check in configuration file, and get the history of your configuration over time for free also It is worth keeping a list of the available configuration options for application in the same repository as its source code

Manging Application Configuration

How does it Works? Cont..



Manging Application Configuration

Modeling Configuration

- Each configuration setting can be modeled as a tuple, so the configuration for an application consists of a set of tuples
- Tuple value typically depends upon:
  - The application
  - The version of the application
  - The environment it runs in (for example, development, UAT, performance, staging, or production)

Use cases when modeling configuration information

Adding a new environment

Creating a new version of the application

Promoting a new version of your application from one environment to another

Relocating a database server

Manging Application Configuration

Principles of Managing Application Configuration

- Keep the available configuration options for your application in the same repository as its source code, but keep the values somewhere else
- Configuration settings have a lifecycle completely different from that of code, while passwords and other sensitive information should not be checked in to version control at all
- Configuration should always be performed by automated processes using values taken from configuration repository, so that one can always identify the configuration of every application in every environment
- Test your configuration scripts
- Use clear naming conventions for configuration options
- Ensure that configuration information is modular and encapsulated
  - So that changes in one place don't have knock-on effects for other, apparently unrelated, pieces of configuration

Manging Application Configuration

Testing Configuration

- **There are two parts to testing configuration:**

The first stage is to ensure that references to external services in configuration settings are good

Example: As part of deployment script, ensure that the messaging bus are configured to use is actually up and running at the address configured

Deployment or installation script should fail if anything application depends on is not available; this acts as a great smoke test for configuration settings

The second stage is to actually run some smoke tests once application is installed to make sure it is operating as expected

Example: This should involve just a few tests exercising functionality that depends on the configuration settings being correct

Managing Environments

Environments ?

No application is Island

Every application depends on hardware, software, infrastructure, and external systems in order to work

Any application depends on:

- Application Configuration like port, messaging bus etc.
- Operating System Configuration

Managing Environments

Problems in Managing Configuration

If the **collection of configuration** information is **very large**
One **small change can break the whole application** or severely degrade its performance
Once it is broken, finding the problem and **fixing it takes an unknown amount of time** and requires senior personnel
It is **extremely difficult to precisely reproduce manually configured environments** for testing purposes
It is **difficult** to maintain such environments without the configuration, and hence behavior, of different nodes drifting apart

Managing Environments

How to Manage Environments

Best Way

fully automated Process

It should always be cheaper to create a new environment than to repair an old one

Managing Environments

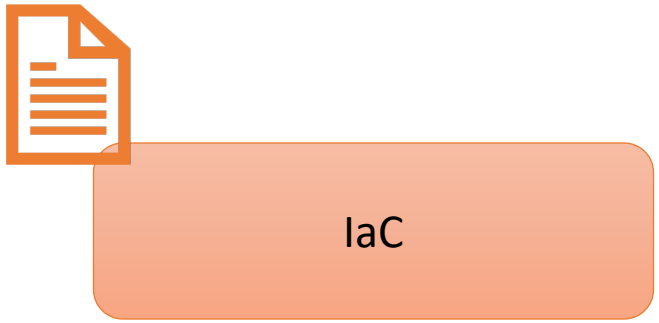
The kinds of environment configuration information should be concerned

- The various operating systems in environment, including their versions, patch levels, and configuration settings
- The additional software packages that need to be installed on each environment to support application, including their versions and configuration
- The networking topology required for your application to work
- Any external services that your application depends upon, including their versions and configuration
- Any data or other state that is present in them (for example, production databases)

Note: There are two principles of an effective configuration management strategy: Keep binary files independent from configuration information, and keep all configuration information in one place

Infrastructure as Code [IaC]

Introduction



Method of writing and deploying machine-readable definition files that generate service components

IaC helps IT operations teams manage and provision IT infrastructure automatically through code without relying on manual processes

IaC is often described as “programmable infrastructure”

Infrastructure as Code [IaC]

IaC for DevOps

- In generic software development, a fundamental constraint is the need for the environment
  - i.e. Testing or Staging environment = Live Environment

Note:
This is the only way of assuring that the new code will not crash with existing code definitions – generating errors or conflicts that may compromise the entire system

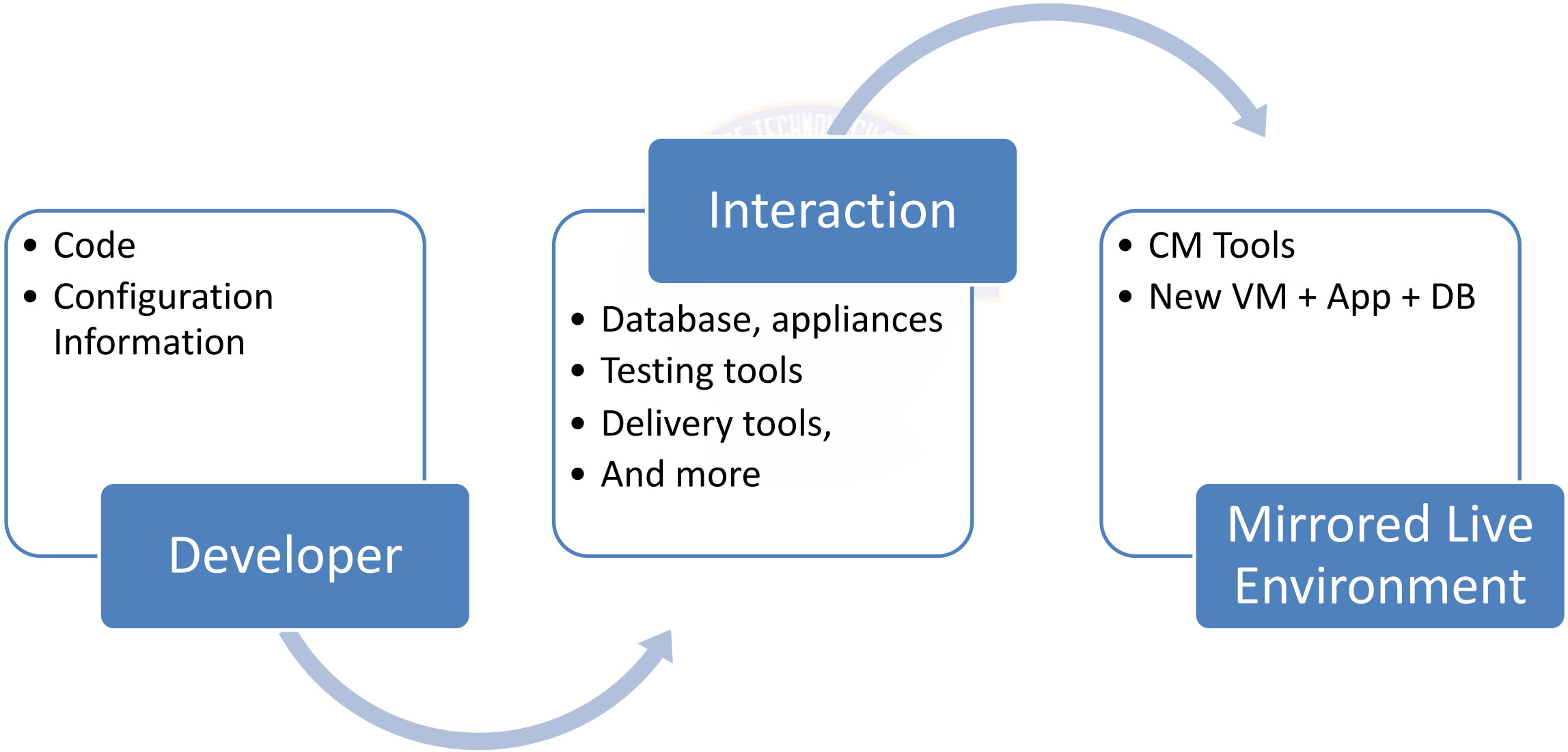
Infrastructure as Code [IaC]

Before IaC and DevOps?

System Admin Team	• Create / Setup Environment • Try to mirror the live environment
DB Admin Team	• Setup / create Database Environment • Handover the machine to Development Team
Development Team	• Deliver the code in above Environment and handover it to Testing Team
Test Team	• Test team run several operational and compliance tests • After all successful process code will hit to Live Production Environment

Infrastructure as Code [IaC]

What is achieved with the help of DevOps and IaC?



Infrastructure as Code [IaC]

Benefits of IaC Summary

Reducing Shadow IT

Improving Customer Satisfaction

Operating Expenses [OPEX]

Capital Expenditure [CAPEX]

Standardization

Safer Change Management

Infrastructure as Code [IaC]

Risks involved with IaC

Missing proper planning

IaC requires new skills

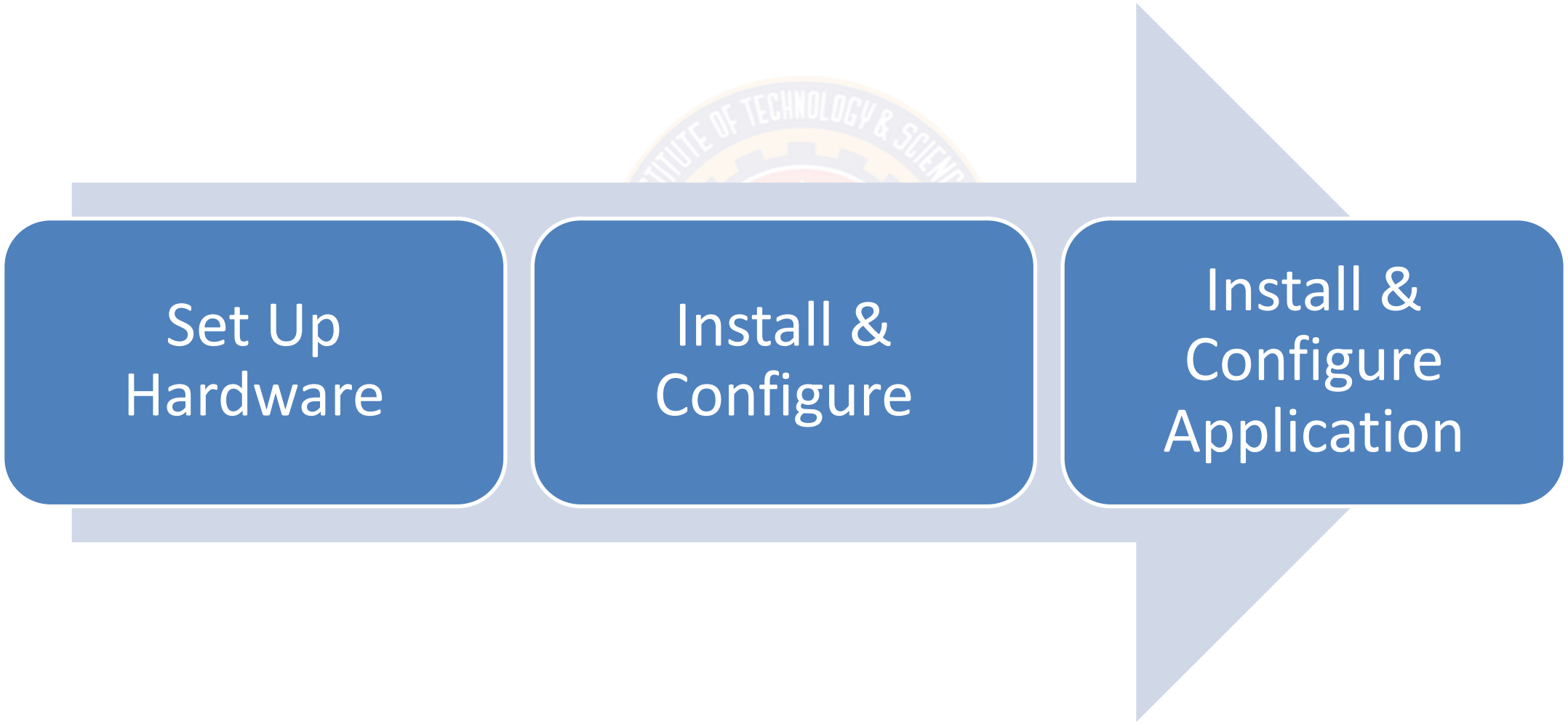
Error replication

Configuration Drift

Accidental Destruction

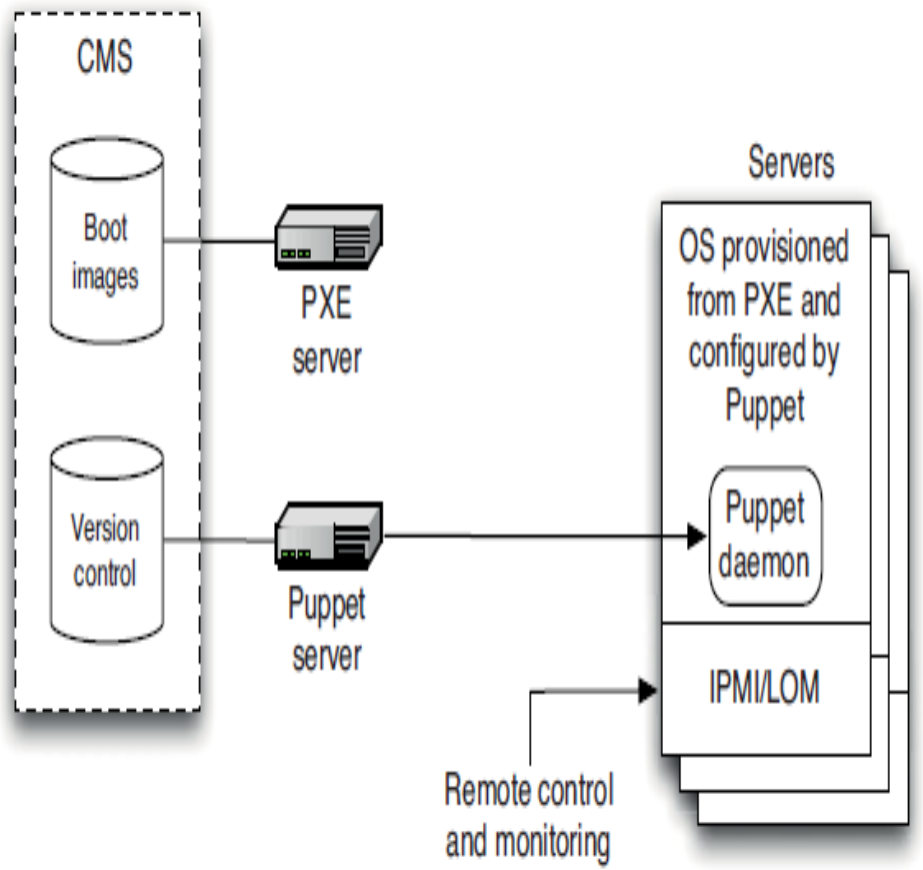
Provisioning Servers

What is Provisioning Servers



Provisioning Servers

An Automate Approach to Provisioning



DHCP

Dynamic Host Configuration Protocol

- Client request for IP
- Respond to the Request

PXE

Preboot Execution Environment

- Request for Boot Image Info
- Boot Image File Name

TFTP

Trivial File Transfer Protocol

- Request file of Boot Image
- Boot Image File to the client

An Automate Approach to Provisioning

Benefits

- Reduction to Man Hours for installation and configuration
- Reduce Operational Cost
- Reduces Errors
- Better Quality Assurance

Configuration management in DevOps

Clear thought

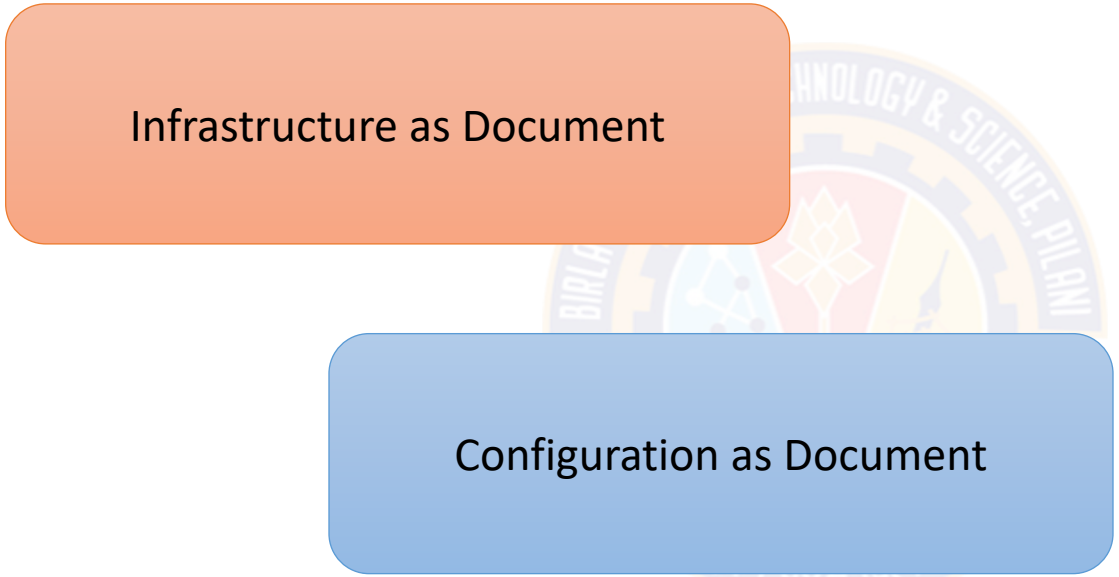
Configuration Management **Vs** **Change Management**

Configuration management deals with the state of any given infrastructure or software system at any given time

Change Management deals with how changes are made to those configurations

Configuration management in DevOps

Traditional Approach



Configuration management in DevOps

New Approach

Infrastructure as code : “It is defining the entire environment definition as a code or a script instead of recording in a formal document”

Configuration as code : “It is nothing but defining all the configurations of the servers or any other resources as a code or script and checking them into version control”

Configuration management in DevOps

Tools

- Puppet
- Chef
- Ansible
- Terraform
- Cloud formation etc.

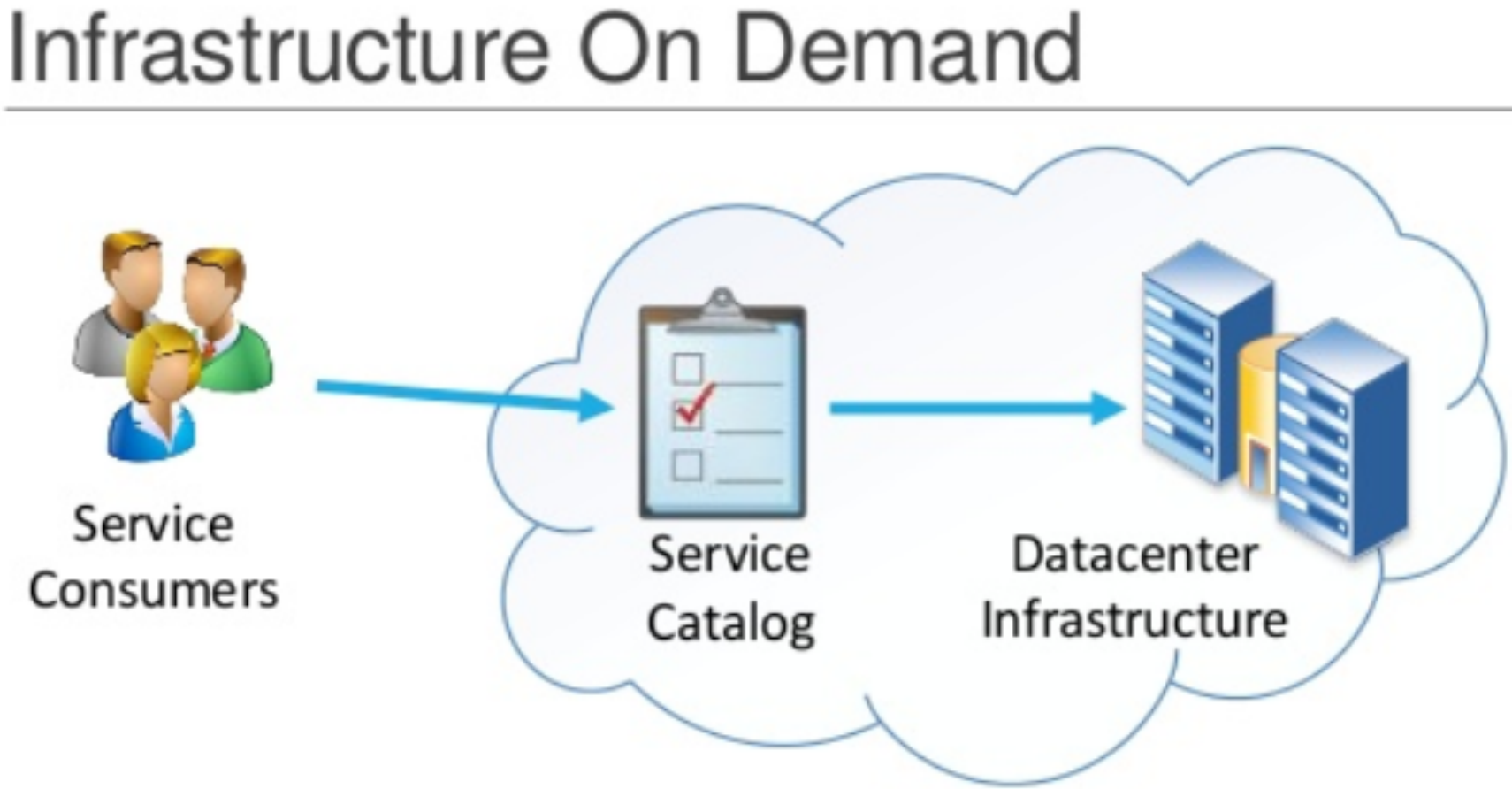
Managing on-demand infrastructure

Why

- We live in a society where you can get anything on demand, right from your smartphone
- You can have groceries delivered to your house
- You can hire someone to clean your home or run your errands
- And ride-sharing services like Uber
- So why should infrastructure be any different?

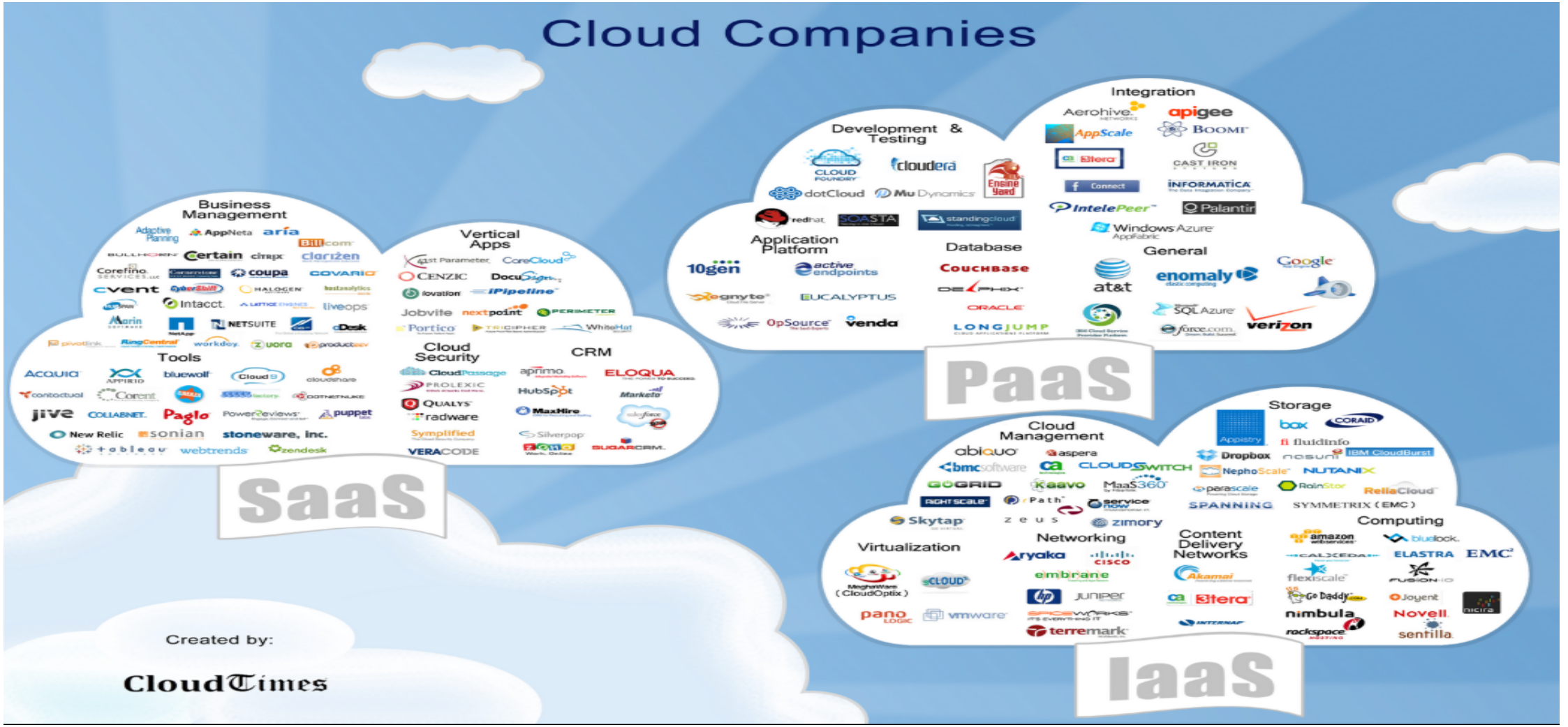
Managing on-demand infrastructure

Cloud Provider



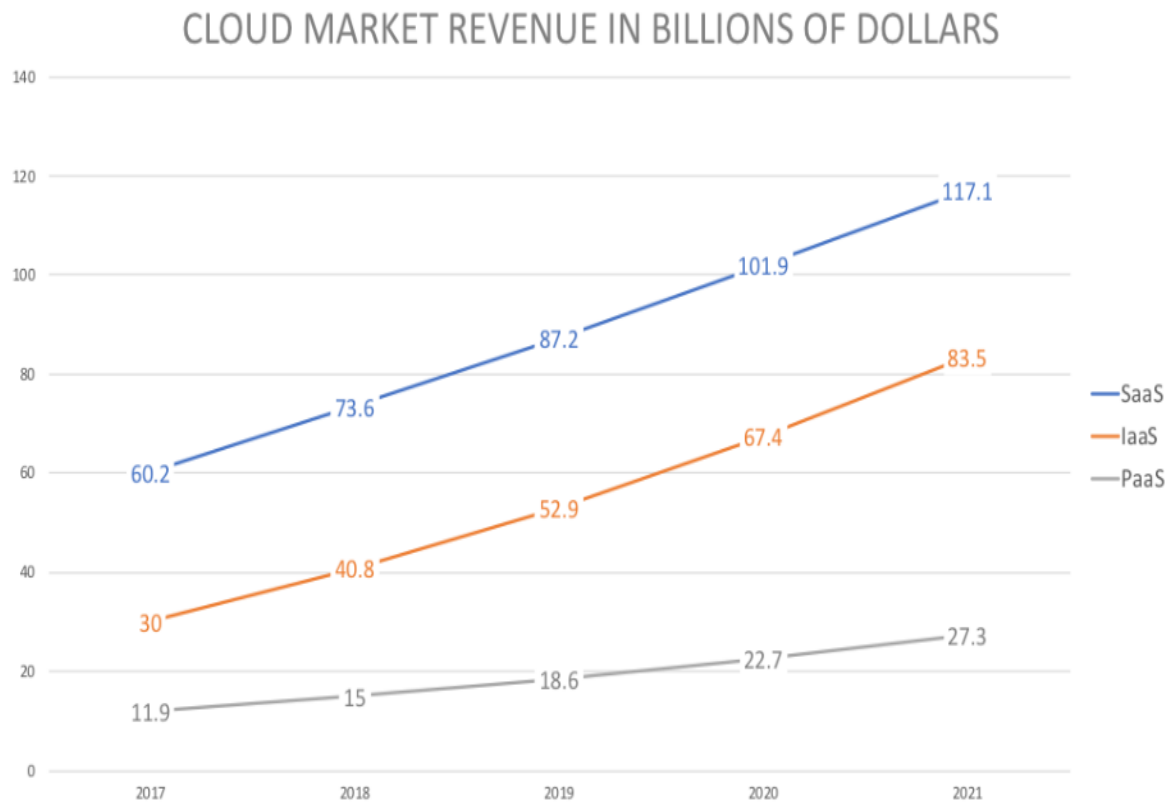
Managing on-demand infrastructure

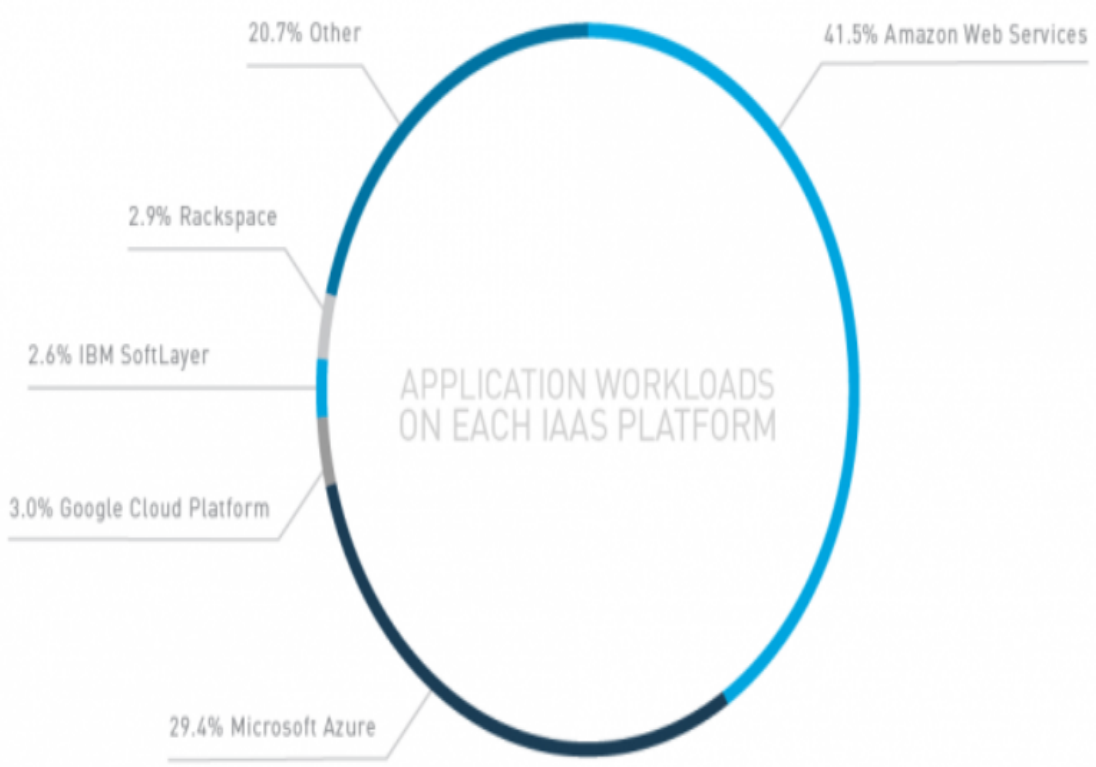
Cloud Companies



Managing on-demand infrastructure

Stats





Managing on-demand infrastructure

Benefits of On-Demand Infrastructure

- Cost savings
- Scalability and flexibility
- Faster time to market
- Support for DR and high availability
- Focus on business growth

Auto Scaling

Auto Scaling Offers

- Auto Scaling typically means allowing some servers to go to sleep during times of low load
- Saving on Power and Energy
- For companies using infrastructure hosted in the cloud, Auto Scaling can mean lower bills, because most cloud providers charge based on total usage rather than maximum capacity
- Auto Scaling can help by allowing the company to run less time-sensitive workloads on machines that get freed up by Auto Scaling during times of low traffic
- Auto Scaling can offer greater uptime and more availability in cases where production workloads are variable and unpredictable

Auto Scaling

Approaches

Scheduled Auto Scaling Approach

This is an approach to Auto Scaling where changes are made to the minimum size, maximum size, or desired capacity of the Auto Scaling group at specific times of day

E.g. E-commerce sites: Flipkart Big Billion Day, Amazon's Great India Sale

Predictive Auto Scaling

The idea is to combine recent usage trends with historical usage data as well as other kinds of data to predict usage in the future, and Auto Scale based on these predictions

E.g. Online Social and Media Hosting

Auto Scaling

Who Offers Auto Scaling?

- Amazon Web Services (AWS) : In 2009, Amazon launched its own Auto Scaling feature along with Elastic Load Balancing
- Netflix is the well know consumer of Auto Scaling at AWS
- Microsoft's Windows Azure : Around 2013, Microsoft announced that Auto Scaling support to its Windows Azure cloud computing platform
- Oracle Cloud Platform: It allows server instances to automatically scale a cluster in or out by defining an Auto Scaling rule
- Google Cloud Platform : In 2015 Google Compute Engine announced a public beta of its Auto Scaling feature for use

References

Text Book Mapping

- Text Book 2: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation by Jez Humble, David Farley. Publisher: Addison Wesley, 2011: Chapter 2 : Configuration Management
- Reference Book1:Effective DevOps: Building A Culture of Collaboration, Affinity, and Tooling at Scale by Jennifer Davis , Ryn Daniels. Publisher: O'Reilly Media, June 2016: Chapter 4 Foundation Terminology and concepts
- https://www.bmc.com/blogs/infrastructure-as-code/

Knowledge Check

Quick recap

- Q.1. Ops feels they don't get enough documentation from dev to deploy new versions of the team's application. Which of the following is a DevOps-friendly solution that might help?

- That the teams collaborate a configuration management tool
- Kick off a more systematic documentation project
- Require documentation before each release
- Make sure a developer is on call for each release

Knowledge Check

Quick recap

- Q.2. What are the major driving forces that are changing the fundamental job of operations?

- ✓ Treating infrastructure as code
- Using your own physical infrastructure
- Using more manual processes
- ✓ Serverless architectures

Knowledge Check

Quick recap

- Q.3. What factors contribute to effectively troubleshooting and isolating problems in the area of operations and deployment? Select all that apply.

- Automated configuration management
- Strong test coverage
- Automated load balancing and failover
- Infrastructure as code
- ✓ All of the above

Q&A

Thank You!

In our next session: