BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

Review CS#7

Continuous Deployment

- Introduction to Deployment
- Deployment Consideration
- Challenges of Deployment
- Deployment pipeline
- Structure of Deployment Pipeline
- Basic Deployment Pipeline
- Stages of Deployment Pipeline
- Human Free Deployment
- Implementing a Deployment Pipeline
- Rolling back deployment
- Zero-downtime Release
- Deployment Strategies

Agenda

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

Continuous Monitoring

Introduction to Monitoring

- The process of observing and recording system state changes and data flows

State changes can be expressed by direct measurement of the state or by logs recording updates that impact part of the state

Data flows can be captured by logging requests and responses between both internal components and external systems

Continuous Monitoring

Monitoring fall into five different categories

1) Identifying failures and the associated faults both at runtime and during postmortems held after a failure has occurred

2) Identifying performance problems of both individual systems and collections of interacting systems

3) Characterizing workload for both short- and long-term capacity planning and billing purposes

4) Measuring user reactions to various types of interfaces or business offerings

5) Detecting intruders who are attempting to break into the system

Continuous Monitoring

What to Monitor

- The data to be monitored for the most part comes from the various levels of the stack

Goal of Monitoring	Source of Data
Failure Detection	Application and Infrastructure
Performance Degradation Detection	Application and Infrastructure
Capacity Planning	Application and Infrastructure
User reaction to business offerings	Application
Intruder detection	Application and Infrastructure

Above Table lists the insights you might gain from the monitoring data and the portions of the stack where such data can be collected

Continuous Monitoring

Fundamental items to be monitored

Inputs

Resources

hard resources such as CPU, memory, disk, and network—even if virtualized

soft resources such as queues, thread pools, or configuration specifications

Outcomes

include items such as transactions and business-oriented activities

Goals Monitoring

Failure Detection

Failures of any element in physical infrastructure is possible

The total failures are relatively easy to detect

No data is flowing where data used to flow

Partial failures that are difficult to detect

Partial failures also manifest as performance problems

Goals Monitoring

Failure Detection

Detecting software failures :

1) The monitoring software performs health checks on the system from an external point
2) A special agent inside the system performs the monitoring
3) The system itself detects problems and reports them

Hardware Failure:

datacenter provider's responsibility

Software Failure:

Dependency Software failure
Software Misconfiguration

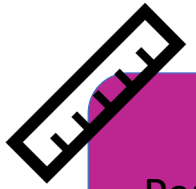
Goals of Monitoring

Performance Degradation

- Degraded performance can be observed by comparing current performance to historical data—or by complaints from clients or end users
- Ideally your monitoring system catches performance degradation before users are impacted at a notable strength

Goals of Monitoring

Performance Degradation



Performance measures

Latency

The time from the initiation of an activity to its completion

It is the period from a user request to the satisfaction of that request

Throughput

The number of operations of a particular type in a unit time

Utilization

The relative amount of use of a resource

Hard resources : CPU (80%), Memory, disk

Soft resources: queues or thread pools

Goals of Monitoring

Capacity Planning

Long-term Capacity Planning

Involves humans and has a time frame on the order of days, weeks, months, or even years

This capacity planning is intended to match hardware needs, whether real or virtualized, with workload requirements

Example:

In a physical datacenter, it involves ordering hardware

In a virtualized public datacenter, it involves deciding on the number and characteristics of the virtual resources

Note: In capacity planning characterization of the current workload gathered from monitoring data and a projection of the future workload based on business considerations and the current workload

Goals of Monitoring

Capacity Planning

Short-term Capacity Planning

Planning is performed automatically and has a time frame on the order of minutes

In this capacity planning the context of a virtualized environment such as the cloud, creating a new virtual machine (VM) for an application or deleting an existing VM

Example:

Monitoring the usage of the current VM instances was an important portion of each option

Note: Charging for use is an essential characteristic of the cloud as defined by the U.S. National Institute of Science and Technology

Goals of Monitoring

User Interaction

User satisfaction depends

The latency of a user request

The reliability of the system with which the user is interacting

User interface modification

Types of User Interaction Monitoring:
Real user monitoring (RUM)
Synthetic monitoring

Goals of Monitoring

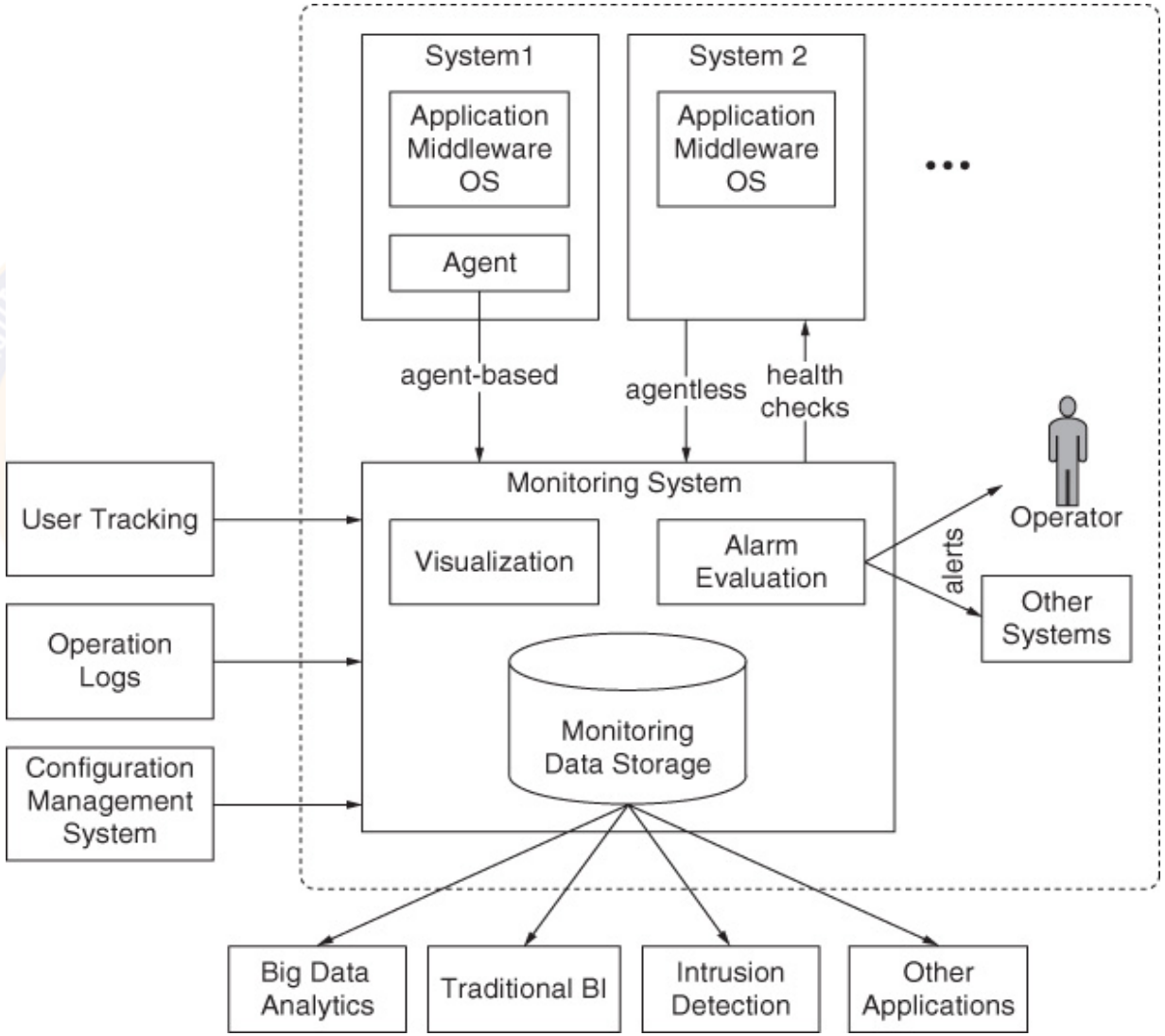
Intrusion Detection

- Intruders can break into a system by disrupting an application
  - Example: Through incorrect authorization
- Applications can monitor users and their activities to determine whether the activities are consistent with the users' role in the organization
- An intrusion detector is a software application that monitors network traffic by looking for abnormalities
- Intrusion detectors use a variety of different techniques to identify attacks:
  - They use historical data from an organization's network to understand what is normal
  - They use libraries that contain the network traffic patterns observed during various attacks
  - Example: Current traffic on network vs Expected traffic in historical data
    - The organization may have a policy disallowing external traffic on particular ports

Continuous Monitoring

How to Monitor?

- Agentless
- Agent-based
- Health Check: External systems can also monitor system or application-level states through health checks, performance-related requests, or transaction monitoring



Continuous Monitoring

Why Monitoring

- A monitoring system allows operators to drill down into detailed monitoring data and logs; which helps in:
- Error diagnosis
- Root Cause Analysis
- Deciding on the best reaction to a problem

Monitoring Operation Activities

Monitoring Operations

- Operations tools monitor resources such as configuration settings to determine whether they conform to pre-specified settings and monitor resource specification files to identify any changes
- Both of these types of monitoring are best done by agents that periodically sample the actual values and the files that specify those values
- There are different Operation Tools like Chef, Puppet, Saltstack and Ansible etc.,
- The offerings of different configuration management tools now available with both Agent Based and Agentless

Monitoring Operation Activities

Collection and Storage

- The core of monitoring is recoding and analyzing time series data, namely, a sequence of time-stamped data points
- Three key challenges in collection and storage:
  - Collating related items by time:
    - Time stamps in a distributed system are not going to be consistent
  - Collating related items by context:
    - If there is any parallel process for the same operation, then it is very difficult to reconstruct a sequence of events to diagnose a problem
  - Handling the volume of monitoring data
    - Big Data, Hadoop etc.

- Change in Monitoring Configuration

Monitoring Operation Activities

Log

sources of the logs

- Applications
- Web servers
- Database systems
- DevOps pipeline
- Another Logs by
  - Operations tools
  - An upgrade tool
  - Migration tool
  - Configuration management tool

Use of Logs

- During operations to detect and diagnose problems
- During debugging to detect errors
- During post-problem forensics to understand the sequence that led to a particular problem

Monitoring Operation Activities

Log

- General rules about writing logs

Logs should have a consistent format
Logs should include an explanation for why this particular log message was produced
Log entries should include context information (Process ID, Request ID, VM ID etc.,)
Logs should provide screening information (Severity level, Alert level)

Monitoring Operation Activities

Graphing and Display

- It is useful to visualize all relevant data collected by monitoring system

Monitoring Operation Activities

Alarms and Alerts

Alerts:

Alerts are raised for purposes of informing

Alerts are raised in advance of an alarm

Example: The datacenter temperature is rising

Alarms:

Alarms require action by the operator

Or

Alarms require action by another system

Example: The datacenter is on fire

Alarms and alerts can be triggered by

Events

A particular physical machine is not responding

By values crossing a threshold

The response time for a particular disk is greater than an acceptable value

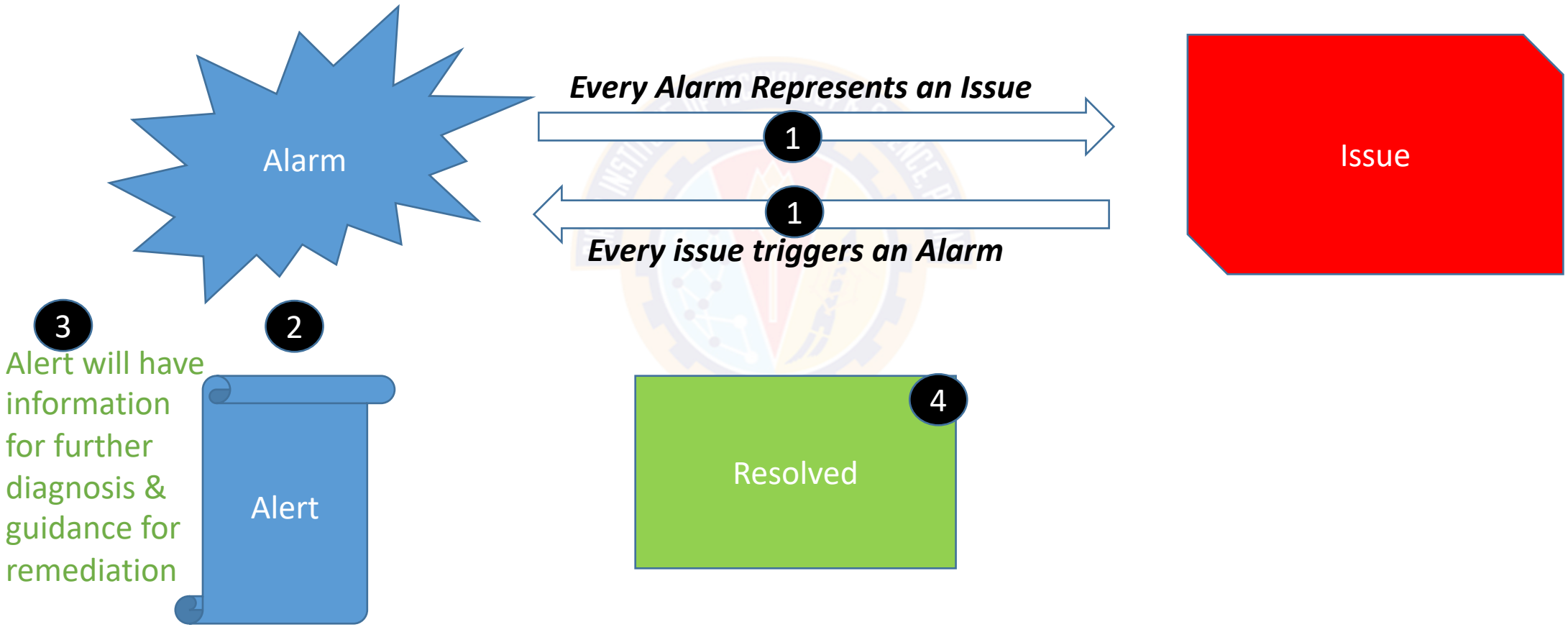
By sophisticated combinations of values and trends

Percentage monitoring of a file system

CPU Utilization on peak in a Day

Monitoring Operation Activities

Alarms and Alerts



Monitoring

Challenges in Monitoring by DevOps

- Challenge 1: Monitoring Under Continuous Changes
- Challenge 2: Bottom-Up vs. Top-Down and Monitoring in the Cloud
- Challenge 3: Monitoring a Microservice Architecture
- Challenge 4: Dealing with Large Volumes of Distributed (Log) Data

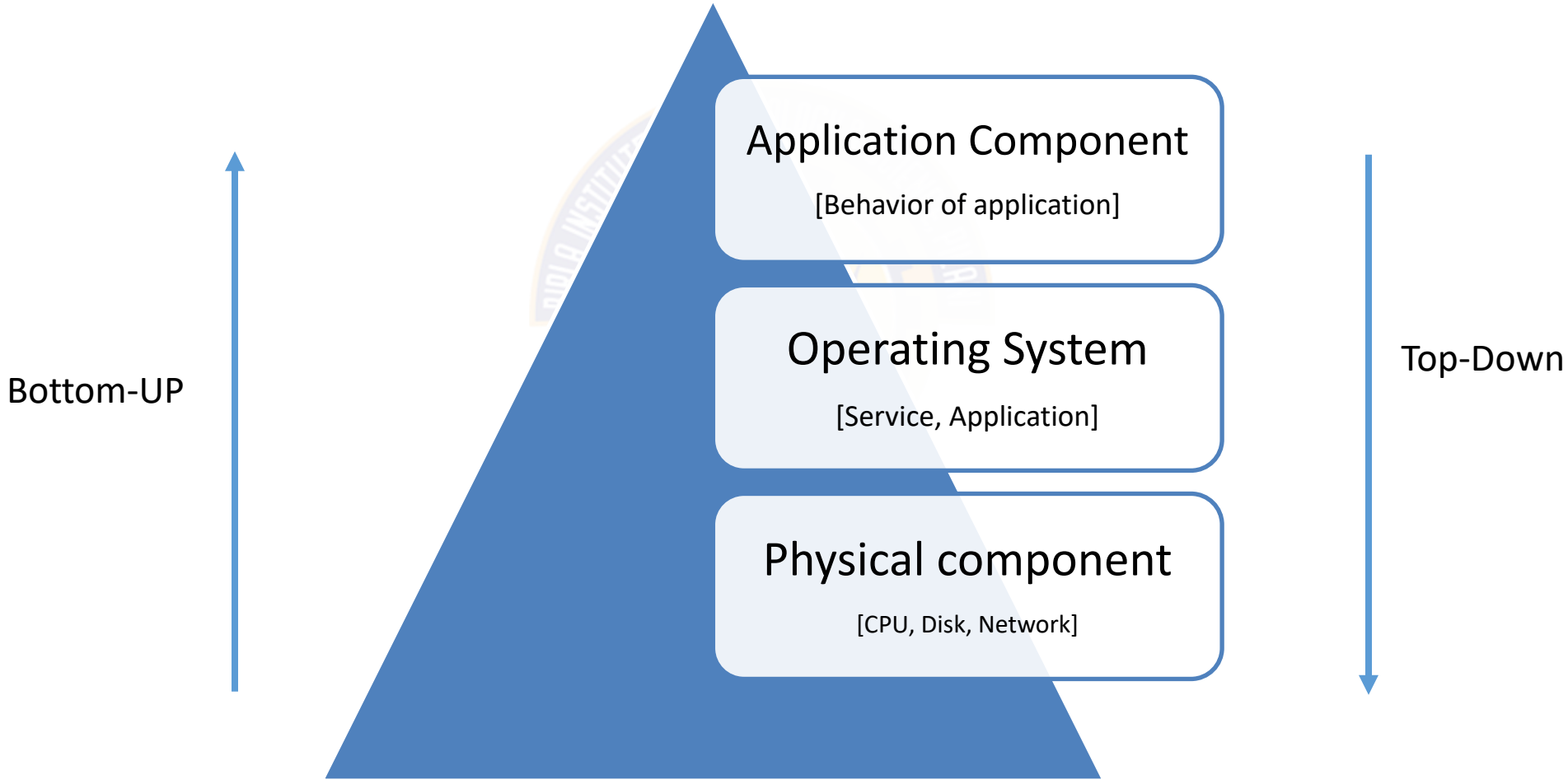
Challenges in Monitoring

Challenge 1: Monitoring Under Continuous Changes

- Here the solution is to automate the configuration of alarms, alerts, and thresholds as much as possible; the monitoring configuration process is just another DevOps process that can and should be automated
- Example:
  - When you provision a new server, a part of the job is to register this server in the monitoring system automatically
  - When a server is terminated, a de-registration process should happen automatically
  - For example, the monitoring results during canary testing for a small set of servers can be the new baseline for the full system and populated automatically

Challenges in Monitoring

Challenge 2: Bottom-Up vs. Top-Down and Monitoring in the Cloud



Challenges in Monitoring

Challenge 2: Bottom-Up vs. Top-Down and Monitoring in the Cloud

- Adopting a more top-down approach for monitoring cloud-based and highly complex systems is an attempt to solve these problems
  - You monitor the top level or aggregated data and only dive into the lower-level data in a smart way if you notice issues at the top level
  - The lower-level data must still be collected but not systematically monitored for errors
- Risk : By above solution you are sacrificing the opportunity to notice issues earlier; and it might already be too late to prevent a bigger impact once you notice that something is wrong at the top level

Note: There is no easy solution, bottom-up and top-down monitoring are both important and should be combined in practice

Challenges in Monitoring

Challenge 3: Monitoring a Microservice Architecture

- Adoption of a microservice architecture enables having an independent team for each microservice
- Every external request may potentially travel through a large number of internal services before an answer is returned
- In a large-scale system, one part or another may experience some slowdown at any given time, which may consequently lead to a negative impact on an unacceptable portion of the overall requests

Note: Need of intelligent monitor systems; one can monitor at microservice level

Challenges in Monitoring

Challenge 4: Dealing with Large Volumes of Distributed (Log) Data

- Operators should use varied and changeable intervals rather than fixed ones, depending on the current situation of the system
  - If there are initial signs of an anomaly or when a periodic operation is starting, set finer-grained monitoring
  - Return to bigger time intervals when the situation is resolved or the operation completed
- Use a modern distributed logging or messaging system for data collection
  - A distributed logging system such as Logstash can collect all kinds of logs and conduct a lot of local processing before shipping the data off
  - This type of system allows you to reduce performance overhead, remove noise, and even identify errors locally

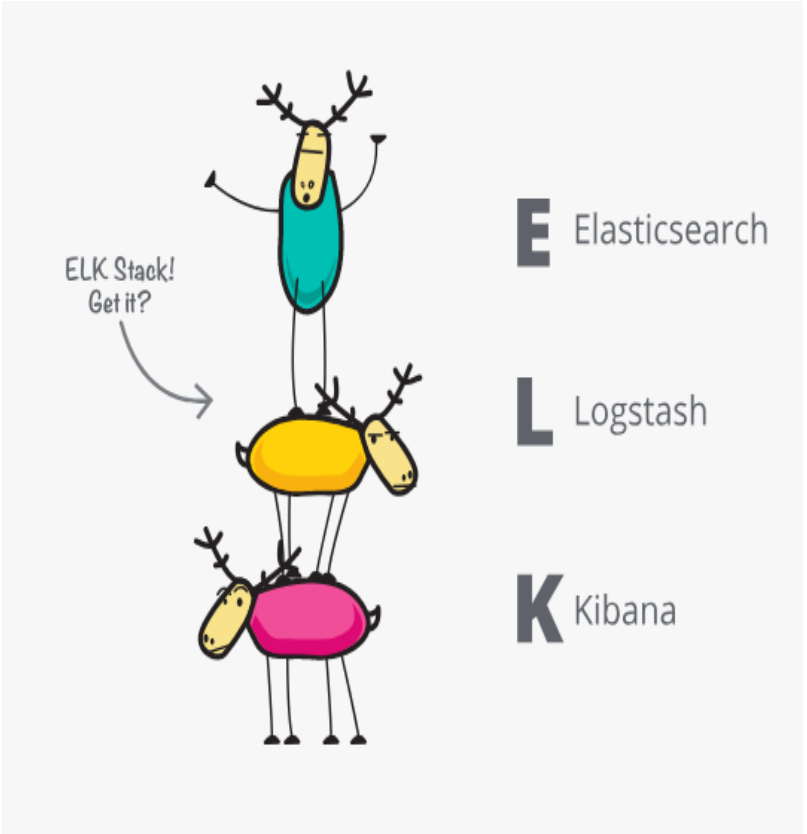
Continuous Monitoring

Monitoring Tools

Continuous Monitoring

ELK

- ELK is the acronym for three open source projects
- Elasticsearch, Logstash, and Kibana
- Elasticsearch is a search and analytics engine
- Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch
- Kibana lets users visualize data with charts and graphs in Elasticsearch



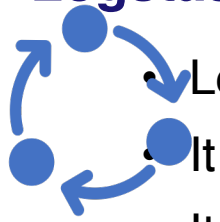
ELK

Elasticsearch

- ELK is started with Elasticsearch
- The open source, distributed, RESTful, JSON-based [Key Pair value] search engine
- Easy to use, scalable and flexible, it earned hyper-popularity among users and a company formed around it
- Elasticsearch lets you perform and combine many types of searches — structured, unstructured, geo, metric
- Elasticsearch aggregations let you zoom out to explore trends and patterns in your data
- Elasticsearch is a NoSQL database that is based on the Lucene search engine

ELK

Logstash

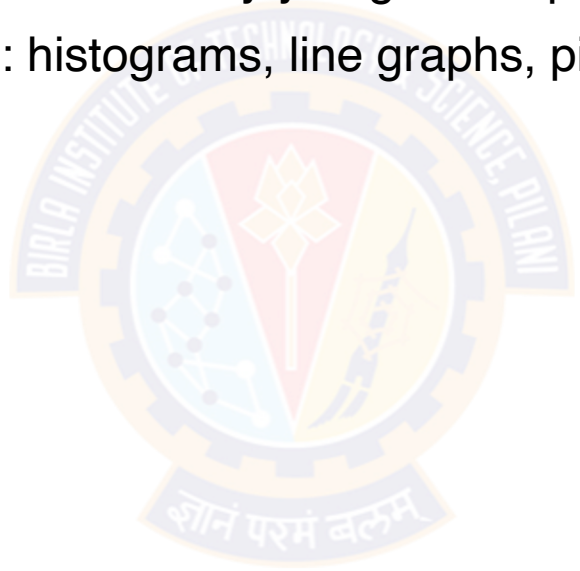


- Logstash is an open source
- It is a server-side data processing
- It is a distributed log management systems, tailored for processing large amounts of text-based logs
- Logstash consumes data from a mass of sources simultaneously, transforms it, and then sends it to your favorite “stash”
- Logstash works in three stages collection, processing, and dispatching

ELK

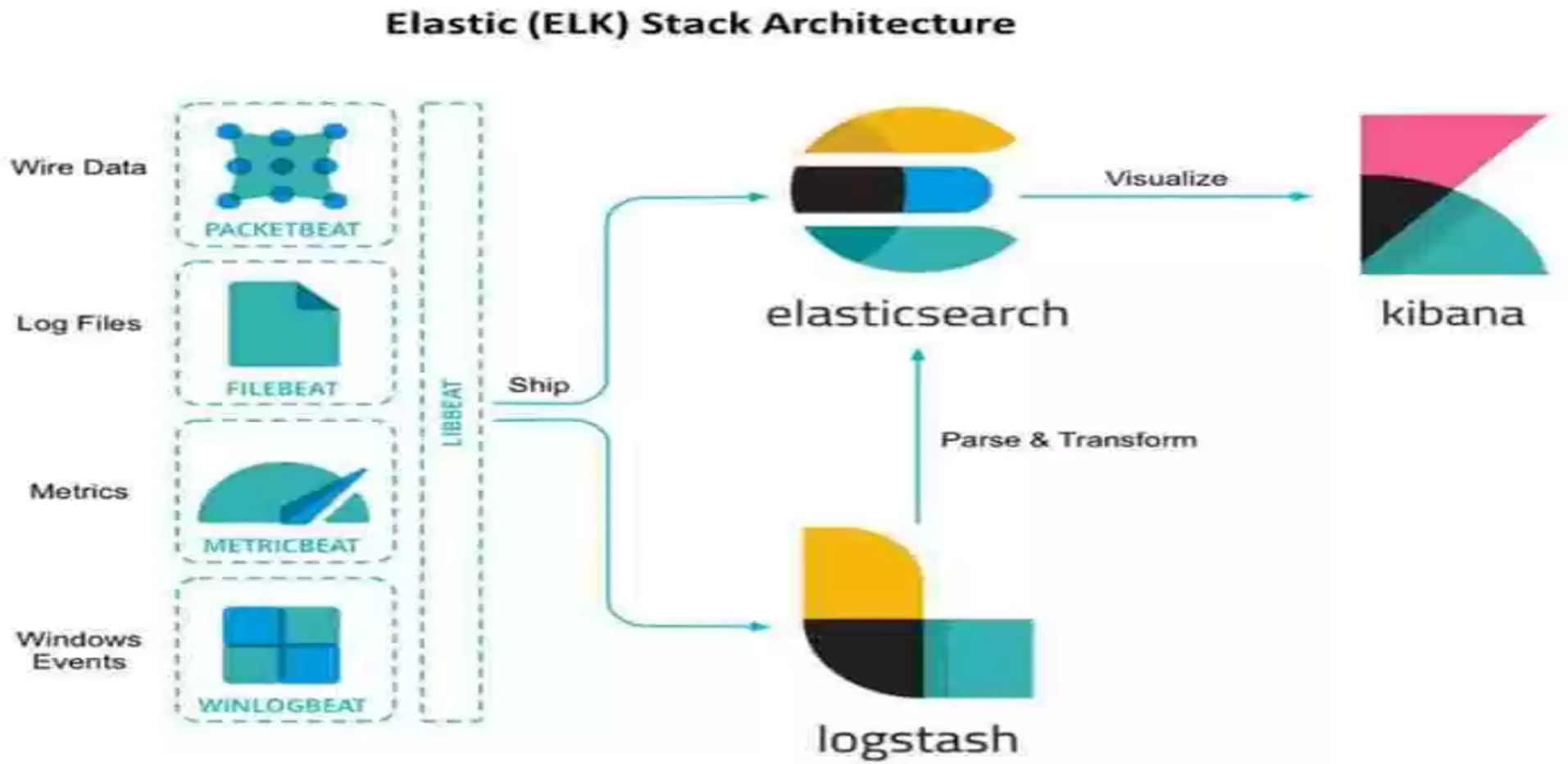
Kibana

- Kibana lets you visualize your Elasticsearch data and navigate the Elastic Stack
- Kibana gives you the freedom to select the way you give shape to your data
- Kibana core ships with the classics: histograms, line graphs, pie charts, sunbursts, and more



ELK

Architecture



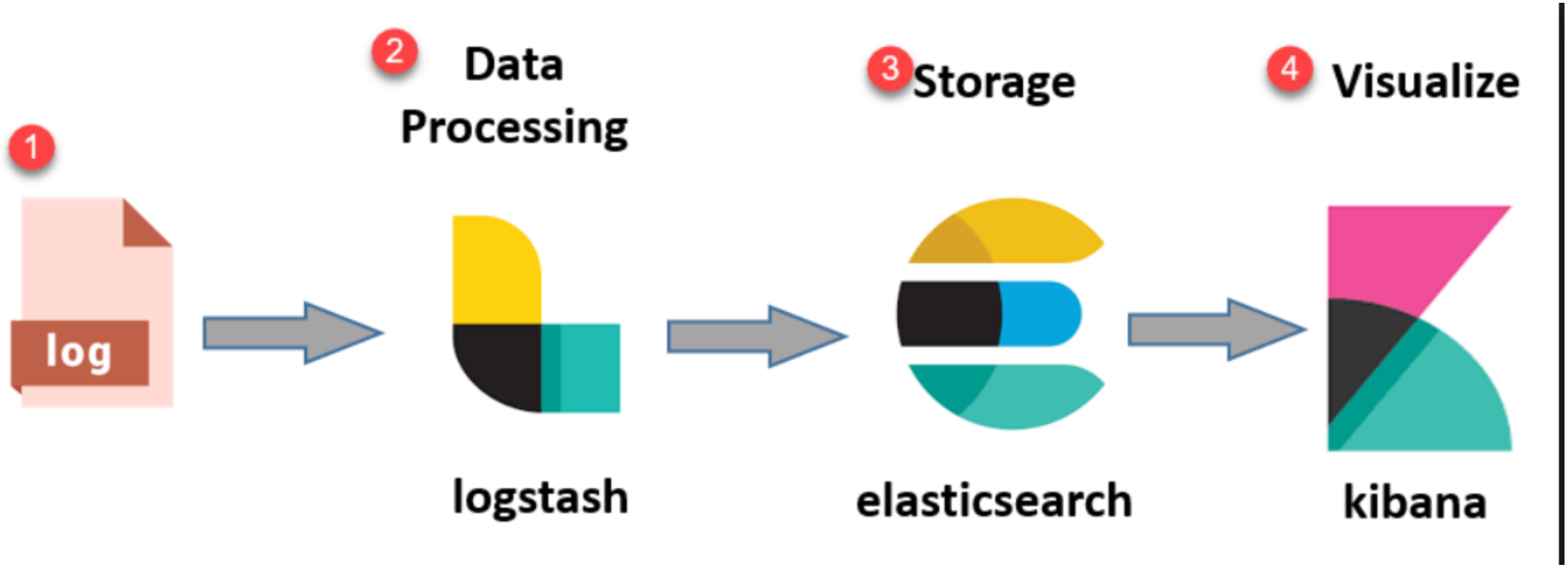
ELK

Features & Benefits

- Security:
  - Protect your Elasticsearch data in a robust and granular way
- Alerting:
  - Get notifications about changes in your data
- Monitoring:
  - Maintain a pulse on your Elastic Stack to keep it firing on all cylinders
- Reporting:
  - Create and share reports of your Kibana charts and dashboards
- Graph:
  - Explore meaningful relationships in your data
- Machine Learning:
  - Automate anomaly detection on your Elasticsearch data

ELK

Quick Review



References

Text Book Mapping

- Text Book 1: DevOps: A Software Architect's Perspective (SEI Series in Software Engineering) by Len Bass, Ingo Weber, Liming Zhu , Publisher: Addison Wesley (18 May 2015) : Chapter 7 : Monitoring

Q&A

Thank You!

In our next session: