DevOps 101 – DevOps Refresher

PUBLISHED ON *May 6, 2018 by Mohd Naeem*

What is DevOps?

- What is **SDLC**?
  - SDLC(Software Development Life Cycle) is an integral part of development, testing, deployment, maintenance of any software.
  - There are many development models to make a project a success (will not enlist all of them, but only two to just give an example)
    - **Water Fall** Model:
      - Project is divided into large chunks of phases e.g Phase 1, 2 and 3 etc
      - Success or failure of current phase has impacts on future phases
      - It is a rigid approach as unless Phase 1 succeeds we can't start phase 2 as there are inter dependencies
      - Releases and Rollbacks are fearsome process and impact the teams success and failure a lot
    - **Agile** model:
      - Project is divided into small sprints and stories (weekly, by weekly or even smaller)
      - There are daily, weekly scrum or status meeting to check the status of the project, bottle necks etc.
      - Releases and Rollbacks are common day to day practice helping project run smoothly.
  - There are two kinds of major groups involved in the life cycle of any project
    - **Developers** (Includes developers, testers, QAs, BAs)
    - **Operations** (includes System/Server admins or web masters, release managers)
- Whats are **Developers**?
  - Developers are responsible for change (functionality) in the state of a project
  - They love to bring as many changes as the business needs.
  - Their productivity is driven by number of changes they bring in the state of the project.
- Whats is **Operations**?
  - are responsible for smooth functioning, releases, deployments of the project
  - They love to keep system stable and love not to bring too many changes to the state of the project to keep a working system stable.
  - Their productivity is driven by less hours of downtime, keeping most of the servers up and running most of the time.
- So **what is wrong** in this approach or model?
  - Developers objective – bring as much as change without caring much about stability.
  - Operations objective – bring stability in the system with as less changes as possible.
  - Both the objectives are going against each other and thus probability of smooth running of project decreases, bottlenecks increase, project is susceptible to failure.
- What is **DevOps**?
  - DevOps is rather a **culture** to resolve above problems.
  - In DevOps culture – which works in more aggressive agile model with small sprints but all changes to the system is driven by **automated builds, deployments, testing, rollbacks.**

- Deployments happen even on hourly basis as the **smallest changes** to the system **goes through automated builds, deployments, testing**, thus chances a change breaking a system is minimized. If some change breaks the automated build, deployment or fails the automated testing is immediately roll-backed to the previous running state.
- Tools and techniques are used to give developers an environment similar using automated setup. So that they can't excuse that something which works on their local systems is not working in staging or production.
- Automated build, deployment and testing tools give the operations the guts to take as many changes as possible because now they don't have an excuse for the stability of the system.
- Thus **DevOps is** = **Dev**(Developers) + **Ops**(Operations) .
  - is a culture or best practices
  - Smaller development cycles
  - More frequent deployments
  - Better collaboration between developers and operations and now both working on same objective:
    - bring quick and stable changes to the system
  - DevOps **is NOT** a standard, tool, or job title.
    - It will use tools to automate the process as much as possible and
    - the tools can vary based on languages, platforms, business needs as
    - their is no standard that if one tool works for say company A will also work for company B.
    - It is more of a best practices based on the needs.
- Salient Features of a DevOps culture:
  1. **Build Automation**:
     - It is the **process of building** the code to make it run using an **automated tool** or script
     - It is independent of the IDE
     - **Benefits** of build automation –
       - **fast**(since no/least manual tasks),
       - **repeatable**(runs the same any time),
       - **consistent**(produces similar results any time),
       - **reliable**(will alert about build errors, will do tasks based on a predefined set of commands in same way) and
       - **portable**(will run o any similar environment the same way)
     - **Tools**:
       - Tool are based on programming language:
         - **Java** – Maven, Ant
         - **JavaScript** – npm
         - **Make** – Unix based
         - **VS** – C#, .NET etc
  2. **Continuous Integration**:
     - It is a continuous process of **merging the developer's code** to the master(deploy-able, release ready branch)
     - It uses the **automated test cases** to pass or fail the developer's change
     - It uses a **CI server** which detects any changes and run the automated test cases against the new build and passes of fails the change.
     - If any developer's code "**Breaks the build**", first they are alerted and the change is **roll-backed** for the developers to fix it.
     - **Benefits**:
       - **Continuations testing**(changes are tested continuously)
       - **Early detection** of problem(due to contentious changes problems are detected early)
       - **No rush** for deployments(developers don't rush to push their code for release)

    - **Frequent releases**(due to contiguous changes there are small but frequent releases)
  - **Tools**:
    - **Jenkins** – open source, widely used to easy integration
    - **TravisCI** – open source, GitHub Integration
    - **Bamboo** – an enterprise product with strong integration with JIRA
3. **Continuous Delivery and Deployment**:
  - Continuous Delivery is process of keeping the **code always in deployable state**
  - Continuous Deployment is the a**ctual process of deploying** the code
  - Both are Not interchangeable terms
  - No standard of how often to deploy. It depends on companies needs
  - **Benefits**:
    - Faster Time to market(due to less problems in the whole process)
    - Less risk(due to increased reliability)
    - Reliable deployments and rollbacks(due to consistent process)
  - **Tools**:
    - Tool
4. **Infrastructure as a code**:
  - The process of using code to **provision and manage resources** or infrastructure by using **code**
  - **Benefits**:
    - **Reusable**(we can execute it as many times)
    - **Scalable**(we can execute it to as many servers
    - **Consistent**(it runs the same ways wherever it runs)
    - **Trackable/Documenting**(any infrastructure changes are well documented as these changes can be committed similar like a normal code)
  - **Tools**:
    - Tool
  - Please see the code below as an example of infrastructure as the code (The below **Dockerfile** uses **Docker hub** to install **Python 2.7**, sets up **working directory**, **copies content** into it and then **runs** the TicTacToe.py python file to run the game):

  ```
  # Use an official Python runtime as a parent image
  FROM python:2.7-slim

  # Set the working directory to /app
  WORKDIR /app

  # Copy the contents to /app container folder
  ADD . /app

  # Run TicTacToe.py when the container launches
  CMD ["python", "TictacToe.py"]
  ```

5. **Configuration Management**:
  - It is the process of **managing or maintaining** the **state of the infrastructure changes** in a consistent, stable and maintainable way
  - We use Infrastructure as a code ensure efficient configuration management

- **Benefits**:
  - **Time saving**(the infrastructure as code can be executed on any numbers of times on any number of servers)
  - **Consistent** (same changes are made wherever the change is executed)
  - **Maintainable**(it will maintainable due to well documentation)
  - **Less configuration drift**(Since similar code is executed there is less configuration drift and that to is well documented)
- **Tools**:
  - **Ansible** – open source, uses YAML config files, does not need a server and agent model, uses declarative configuration
  - **Puppet** – open source, needs a server and agent model, uses declarative configuration
  - **Puppet** – open source, needs a server and agent model, uses procedural configuration
  - **Salt**– needs a server and agent model, uses declarative configuration, uses YAML config files

6. **Orchestration**:
   - It is a process of using a builder tool which automates the whole workflow or process.
   - E.g. docker-compose, Kubernetes etc
   - **Benefits**:
     - **Scalibility**(the orchestration tool can be used to execute the changes to any number of servers)
     - **Stable**(the changes are stable as always executed in the same fashion)
     - **Self servicing**(it is a auto healing method)
     - **Granularity**(there is full control over the whole process due to each steps well defined)
     - **Time saving**(automation leads to quick turn around time)
   - **Tools**:
     - **Kubernetes** – biggest hit these days
     - **Docker-compose**
7. **Monitoring**:
   - It is a process **monitoring the state of a system**, alerting any change in the state, presenting the state of the change in a meaningful manner
   - You can monitor the system resources like CPU, memory, I/O, Network, Logging etc
   - **Benefits**:
     - **Fast recovery** from failures(automated alerts help in recovering from failures by provisioning resources based on the alert)
     - **Automated alerting** and response(the alerts give the impulse for auto healing systems)
     - **Root cause** and visibility(helps in root cause analysis based on the information tracked and monitored)
     - **Auto healing**(With proper health checks and alerting system, we can configure the system to auto heal in the event of failure of a few servers by provisioning more servers)
   - **Tools**:
     - AppDynamics
     - Newrelic

- Here is a website with the links and details of most of the DevOps tool, you click on these links to see the details of each of these. That is why although I enlisted quite a many tools but did not gave much details about them as you can see basic details in the periodic table itself and some of the tools I will cover in details- **https://xebialabs.com/periodic-table-of-devops-tools/ (https://xebialabs.com/periodic-table-of-devops-tools/)**

CATEGORIES DEVOPS •