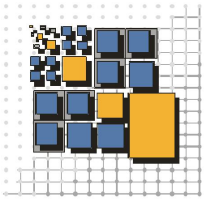

University of Bamberg

Distributed Systems Group

Project Report

With the topic:

CI-CD CLOUD NATIVE

Submitted by:

Group 1

Mohammed Mehedi Hasan
Md Anisul Haque
Md Saiful Ambia Chowdhury
Shoaib Bin Anwar
Rahat Rafiq

Supervisor:
Johannes Manner
Sebastian Böhm

Examiner:
Prof. Dr. Guido Wirtz

Date of submission:
May 23, 2021

Contents

List of Figures

List of Tables

Listings

Abbreviations

Part I

Introduction

1 Introduction

CI-CD means continuous integration and continuous delivery or deployment. This process is trigered when a code is commited into the version control system. A build tool is used on the CI-CD to create the artificates. The next step is to run some unit and integration tests to find out any bugs. Afterwards, passing all the tests the new version of the software is deployed on the desired environment. Otherwise The developers will work on the bugs and fixing those until it is on deployable states. By this way when the final relaese date will come the software can be released smoothly and faster.

Now-a-days, CI-CD can be setup on cloud environemnt which incorporates the main process of CI-CD (build, test, delivery and release) and additionally provides some features flexibility for the CI-CD. One of the main upgrade is software development and release team do not need to worry about the environemnt specification and different tools setup on that environment as well.

In software release there are some manual processes are used. Some processes are manually configure the software dependency on the production environemnt, deploying the software on the invironemnt as well. This processes are more complicated due to sparse environment releated issues and so on. Furthermore, this processes often indicate missing bugs in software which are found in the phase which is rather risky to fix at that time. And doing this rush also introduce more bugs into the system and in the end team members have to work on a lot of stess and pressure as well. But if these processes are made automate in the very first of phases than these kind of issues can be reduced a lot in the begining. And thus CI-CD becomes a growing and interesting concept in the software release process. It is integrate from the begining so whenever a commit is made by the developer the build is done and then test cases are executed and based on the results it was decided whether this particular version will be released or not. if it was not ok then developer should work on the fix and afterwards the fix will be committed. The CI will do the process again and then upon success of the CI it will deplyed on the desired environemnt as well. This automate deployment in the environment is called CD. Using these two methodologies, CI-CD, together make the system more smooth, bug free and moreover stress free software release process. Furthermore, using CI-CD on cloud gives additional edges for the team reagrding software release steps. Especially, setting up the environemnt and scalling the projects are more flexible and easier. In this report we will dive into more in the CI-CD steps and different tools structure.

Part II

CI-CD

1 CI-CD (Continuous Integration-Continuous Deployment)

1.1 Definition

One of the best practices for devops teams to follow is the CI/CD pipeline, which allows them to deliver code updates more frequently and efficiently.

Continuous integration (CI) and continuous delivery (CD) are two concepts that describe a culture, set of operating principles, and set of practices that allow application development teams to produce code changes more frequently and consistently. The CI/CD pipeline is another name for the implementation.

Development teams can develop, prototype, deploy, and run applications in a continuous process using automation and virtualization. This is faster than releasing big batches of updates and patches all at once. It also ensures that apps are published faster, are more up-to-date, and have improved security and scalability, all of which are important aspects of agile networking.

"Continuous integration is a coding philosophy and set of practices that drive development teams to implement small changes and check in code to version control repositories frequently. Because most modern applications require developing code in different platforms and tools, the team needs a mechanism to integrate and validate its changes."[1]

1.2 Components

A CI/CD pipeline is made up of distinct subsets of tasks that are organized into pipeline components. The following are most common pipeline components:

1.2.1 Development

The component at which the application is put together. For possible deployment, source code must be built and packaged. To automate this step of the CI/CD pipeline, a variety of continuous integration tools are available. Since the deployable units are language-dependent, the build tools for the languages used in the deployable units must be named and performed. For example, in JAVA tools like Maven or Gradle are necessary to build a JAVA distribution. A packaging phase may include the automated build elements. Taking the JAVA example a step further, if a Docker Image of the JAVA app is needed, the required Docker Compose steps must be called. Build-centric tests, such as unit tests

[1] https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html

and dependency scanning, can be performed in the build components.[2]

Compilation is required for programs written in languages like Java, C/C++, or Go, while Ruby, Python, and JavaScript do not. Cloud-native software is usually deployed with Docker, so this stage of the CI/CD pipeline creates Docker containers, regardless of the language. Failure to pass the construct stage indicates a fundamental problem.[3]

1.2.2 Testing

This is the section where the code is put to test. Automation will save both time and effort in this situation. Enhancement of personal trust is a big aim of most pipelines. Running tests is the standard method for imparting trust in apps. Test elements come in a variety of sizes and shapes. CI/CD pipelines are natural places to perform experiments as consistency gates as test methodologies evolve. Tests that include the whole application, such as integration tests, soak tests, load tests, and regression tests, are natural fits in addition to build-centric tests. Modern research methods, such as Chaos Engineering, can also be used at the infrastructure level.[4]

This is an overnight procedure in which practical checks, security scans, and consistency tests are run on the software's most recent successful build. The new containers are installed in the continuous testing environment using the most recent docker images prior to the test execution. As a condition for testing, the Kubernetes cluster's persistent volumes will be restored. It's worth noting that all of these tasks are pre-planned and fully automated. The next morning, prior to the regular stand up meetings, the test report is reviewed. The quality assurance team would address any scripting problems, and the production team would fix any programming issues. CT failures are a top priority, and they will be addressed as soon as possible.[5]

1.2.3 Release

Release components are the individual implementations that help achieve Continuous Implementation. If you have a rolling, blue-green, or canary deployment in mind? The orchestration will be handled by release elements in your CI/CD pipeline.[6]

- **Rolling Deployment** : A rolling rollout is a release technique that updates operating instances in a sequential manner. To elaborate, the old framework version is decommissioned, and a new one is installed in its place before all nodes in the series have been replaced.
- **Blue-Green Deployment** : A blue-green deployment is a risk-averse release technique. With two concurrent models of development going, the new release (blue) will gradually overtake the stable version (green), with the stable version remaining

[2]https://harness.io/blog/continuous-delivery/ci-cd-pipeline/
[3]https://semaphoreci.com/blog/cicd-pipeline
[4]https://harness.io/blog/continuous-delivery/ci-cd-pipeline/
[5]https://cinglevue.com/how-to-build-an-efficient-ci-cd-pipeline/
[6]https://harness.io/blog/continuous-delivery/ci-cd-pipeline/

operational until it is considered safe to repurpose or decommission it. Rollbacks are much simpler when blue-green deployments are used.

- **Canary Deployment** : A canary deployment is an evolutionary update approach in which a new upgrade (the canary) is gradually rolled out before the current version is replaced. Canary deployments are carried out in stages. For eg, the first phase would swap 10% of the nodes, and if successful, the second phase would swap 50% of the nodes, and the third phase would swap 100% of the nodes. The stability they offer during a release, as well as the fact that they use less resources than a blue-green deployment, are the key reasons for implementing canary deployments. Canary deployments, on the other hand, can be complicated due to the validation needed to promote canaries.

1.2.4 Deployment

The implementation is simplified since most of the hard work has already been completed in the previous three stages. A release can be completed at any time, with the only qualification being a successful CT period. The scripts for the release must have the following properties:

-The related version number should be added to the docker images.

-The source repositories should be tagged with the version number.

The release will now be implemented in the release pipeline's other environments. The decision to promote the release to the production will ultimately be a business decision. The deployment phase would be simplified with a docker + kubernetes configuration, and the results would be consistent in all environments.[7]

1.3 CI-CD on Cloud

Although cloud computing has multiple meanings, the most basic definition is a framework that allows and facilitates the provisioning of resources. As a result, it can be represented as code or models, making it easier to create repeatable procedures. DevOps' core philosophy is to automate as much as possible the processes/tasks in the software development life cycle.

The efficient structure of cloud computing is one of the most significant advantages of cloud for CI/CD. This is ideal for CI/CD workloads, which are ephemeral and burst. Cloud services can scale up and down dynamically in response to CI/CD workloads. For enterprises, this provides significant management and cost savings. Enterprise firms do not need to run their own servers, but they do need to scale up as CI/CD workloads grow, and they don't want to spend server resources when they're not in use.

A public cloud is one in which the cloud provider hosts all the software and data for the enterprise. Employees who work with a business that uses public cloud can access the applications with only an internet connection. When an organization's computing,

[7] https://cinglevue.com/how-to-build-an-efficient-ci-cd-pipeline/

compute, and networking services are located in a provider's data center, it is referred to as a private cloud. When it comes to operating software, companies choose private cloud for security and storing extremely sensitive data.

The hybrid cloud is a more modern approach that combines private and public cloud services. Organizations may make changes based on traffic and demand considerations. Some consumers may have their own CI/CD systems on their premises. In a hybrid setup, any of these CI/CD systems may extend their workloads to the cloud. This would encourage them to reap the benefits of cloud computing without having to make a full migration.

Workloads can be run in the public cloud when there is a lot of demand, and then returned to the private cloud when things calm down. This method decreases the amount of money spending on cloud services. Furthermore, classified documents, records, and essential programs can all be stored in the private cloud. Less confidential data and programs, on the other hand, can be maintained and run in the public cloud.[8]

1.4 Different Tools for CI-CD on Cloud

Different Tools for CI-CD on Cloud:

- Azure DevOps
- AWS
- Google Cloud

1.4.1 Azure DevOps

Microsoft's Azure DevOps framework is a Software as a Service (SaaS) platform that offers a complete DevOps toolchain for designing and deploying software. It also interfaces with the majority of popular tools, making it an excellent option for orchestrating a DevOps toolchain.

Despite its late October 2018 debut, Azure DevOps is not a newcomer to the DevOps scene. Its ancestors can be tracked all the way back to the 2006 release of Visual Studio Team System. Microsoft has over 80,000 internal customers for this advanced platform with a diverse feature set.

Azure Devops is not only designed for organizations that use Microsoft or Windows end-to-end. Azure DevOps is a complete tool that allows you to be:

Flexible

There's no need to go all-in on Azure DevOps. Any of the services can be adopted separately and integrated into the current tool-chain.

[8] https://devops.com/cloud-and-devops-ci-cd-and-market-analysis/

Platform agnostic

It is crafted to run on every computer (Linux, MacOS, and Windows) and with any language (including Node.js, Python, Java, PHP, Ruby, C/C++,.Net, Android, and iOS apps). Azure DevOps isn't just about companies who develop and launch software.

1.4.2 AWS

AWS CodePipeline is a professional automated continuous distribution service that facilitates automating the release pipelines for fast and stable device and infrastructure updates. Based on the defined release model, CodePipeline automates the develop, test, and deploy phases of the release process whenever there is a code update. This makes possible to offer features and upgrades quickly and consistently. AWS CodePipeline can be seamlessly integrated with third-party platforms like GitHub and many custom plugin. users just have to pay for what they use with AWS CodePipeline. There are no hidden costs or long-term obligations.

Advantages

- Rapid delivery
- Configurable workflow
- Get started fast
- Easy to integrate[9]

1.4.3 Google Cloud

In the public cloud industry, Google Cloud Platform is one of the most common cloud providers. It offers a variety of managed services, and it makes sense to use Google Cloud's managed CI/CD software if you're solely using Google Cloud.[10]

One of the core principles of the google cloud platform is its commitment to open-source development support. A lot of the features of GCP are developed by the community and are easy to integrate in the GCP Platform. Which makes it easier to get support for an application. Even though Google got into the cloud market later than its counterparts but their commitment to open-source development is helping them grow their offered features exponentially.

[9] https://aws.amazon.com/codepipeline/
[10] https://medium.com/swlh/how-to-ci-cd-on-google-cloud-platform-1e631cded335

Part III

Azure DevOps

1 CI-CD on Azure DevOps

1.1 Introduction

Microsoft leveraged its constantly-growing worldwide network of data centers to create Azure, a cloud platform that facilitates building, deploying, and managing services and applications, anywhere.

At its core, Azure is a public cloud computing platform—with solutions including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) that can be used for services such as analytics, virtual computing, storage, networking, and much more. These services can be used to supplement an on premise server or totally replace it.

A CI-CD architecture is implemented in Microsoft Azure as a culmination of two azure services. Those are

- Azure Portal
- Azure DevOps

The steps or tasks of a CI-CD pipeline are configured in Azure DevOps service and the application is deployed in a resource inside Azure Portal. However, there is also options to deploy the application in other places such as AWS, GCP or even on-premise server.

1.2 Azure Portal

The Azure portal is a web-based, graphical user-interface console that works as an alternative to cli tools. With the help of Azure portal, it's possible to manage all the resources in an Azure subscription. For instance, a user can build, test and deploy everything from simple web applications to complex cloud deployments. It also provides custom dashboards for an organized view of resources and tool like azure monitor to inspect the performance of an application.

The main strength of Azure portal is continuous availability. It has an instance available in every Azure data-center, which makes the Azure portal immune to individual failures. The Azure portal also updates continuously and requires no downtime for maintenance activities.[11]

[11] `https://docs.microsoft.com/en-us/azure/azure-portal/azure-portal-overview`

1.3 Azure DevOps

The Azure DevOps platform is a platform created by Microsoft as a software-as-a-service(SaaS) that provides a tool-set with an end to end DevOps environment for developing, integrating and deploying software on a continuous integration and a continuous deployment pipelines. It promotes a culture and set of practices that bring all stakeholder together to complete software development and release process.

Moreover it integrates with almost any industry leading DevOps tools available in the marketplace, such as Docker Registry, Kubernetes, Auzre portal, Terraform, Ansible, GitHub etc.[12]

1.4 Azure DevOps Key Services

The Azure DevOps platform consists of a few key services such as:[13]

- Azure Boards
- Azure Repos
- Azure Pipelines
- Azure Test Plans
- Azure Artifacts

1.4.1 Azure Boards

Azure boards is a service of Azure DevOps through which it is possible to manage a project throughout the entire development life-cycle. It provides the functionality to track tasks, work status, user stories, backlogs, features, and track bugs and defects that are noticed in the project. The three basic work items of azure boards are Epics, Issues, and Task.[14]

1.4.2 Azure Repos

Azure DevOps Repos are a set of repositories that provides the functionality of version control and managing a projects code. It helps to work and coordinate code changes across team. It will allow users to to monitor code, solutions, builds, commits, pushes, Pull requests and branching information about projects.

Azure Repos provides two types of version control:

- Git : distributed version control
- Team Foundation Version Control : centralized version control

[12] https://www.devopsgroup.com/insights/resources/tutorials/all/what-is-azure-devops/
[13] https://www.simplilearn.com/azure-devops-article
[14] https://www.c-sharpcorner.com/article/azure-devops-boards/

1.4.3 Azure Pipelines

Azure Pipelines is a service that caters the need for creating pipelines on Azure Cloud Platform so that it's possible to build, test and deploy code project automatically. There are three key distinct advantages of using Azure DevOps pipelines:[15]

1. Version Control System : Azure Pipelines integrates with GitHub, GitHub Enterprise, Azure Repos Git, TFVC, Bitbucket Cloud and Subversion.
2. Language and application types : Azure Pipeline is compatible with most application types and languages, such as Java, JavaScript, Node.js, Python, .Net, C++, Go, PHP, etc.
3. Deployment Target : Azure Pipelines can be used to deploy project code to multiple targets. such as, container registries, virtual machines, Azure services, or any on-premises or cloud target.

Azure Pipeline is based on a few concepts, which are represented by a few keyword. Such as:

- Pipeline: A workflow that defines how test, build, and deployment steps are run.
- Stage: It can be used to mark the separation of concerns. Each stage contains one or more jobs.
- Job: A stage can contain one or more jobs. Each job runs on an agent.
- Step: The smallest building block of a pipeline. It can either be a script or a task.
- Agent and Agent pools: An agent is an install-able software that runs one job at a time. Agent pool handles multiple agents.
- Artifact: It is a collection of files or packages published by a run. The Artifact is made available to subsequent tasks, such as distribution or deployment.
- Trigger: Trigger tells the pipeline when to run.
- Environment: It is a collection of resources, where the application is deployed.
- Runs: It represents a single execution of a pipeline and collects the logs associated with running the steps and the results of running tests.

[15] `https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops`

1.4.4 Azure Test Plans

Azure Test Plan is a browser-based test management solution for exploratory, planned manual, and user acceptance testing. It provides a browser extension for exploratory testing and gathering feedback from stakeholders. Test Plans are an incredible place for a development team to do their manual testing.[16]

For automated testing as part of the CI/CD workflow, Azure Pipelines is the best option. However, Manual and exploratory testing are still significant part of evaluating quality of a product. Azure Test Plans in Azure DevOps provides three main types of test management artifacts:

1. Test plans : Group test suites and individual test cases together
2. Test suites : Group test cases into separate testing scenarios within a single test plan.
3. Test cases : Validate individual parts of the code or app deployment. It can be ensured that the code works correctly, has no errors, and meets business and customer requirements.

1.4.5 Azure Artifacts

Azure Artifacts is an extension that facilitates the discovery, installation and publication of NuGet, NPM and Maven packages in Azure DevOps. It's highly incorporated with other hubs like Build so that package management can become a smooth part of existing workflows.[17]

1.5 DevOps Workflow Diagram

In Azure DevOps developers manage and keep track of their tasks and backlogs by using the azure boards service. The source code is developed in a local environment. Azure provides seamless integration with most of the popular IDEs like VS Code, Intellij Idea and Eclipse through extension. Whenever a new commit is pushed to the Azure Repos service it triggers a new build job at Azure Pipelines. A build job can have many stages like installing tools, testing and more. These steps ion a build job can be defined manually by a developer in a azure CI pipeline. After the build is complete, the generated application artifacts are stored in azure artifacts service. Continuous Delivery pipeline is also configured in the azure pipeline service. Whenever a CD pipeline is triggered, it fetches all the necessary artifacts from azure artifacts service and it deploys the application in a pre-configured environment, this environments are used to define and isolate development, QA and production stages in a CI-CD pipeline. These environments are actually stored as a resource groups in azure portal.[18]

[16] https://docs.microsoft.com/en-us/azure/devops/test/create-a-test-plan?view=azure-devops

[17] https://azure.microsoft.com/en-us/services/devops/artifacts/

[18] https://www.veritis.com/wp-content/uploads/2019/02/azure-devops-ci-cd-pipeline-flow-veritis-1024x

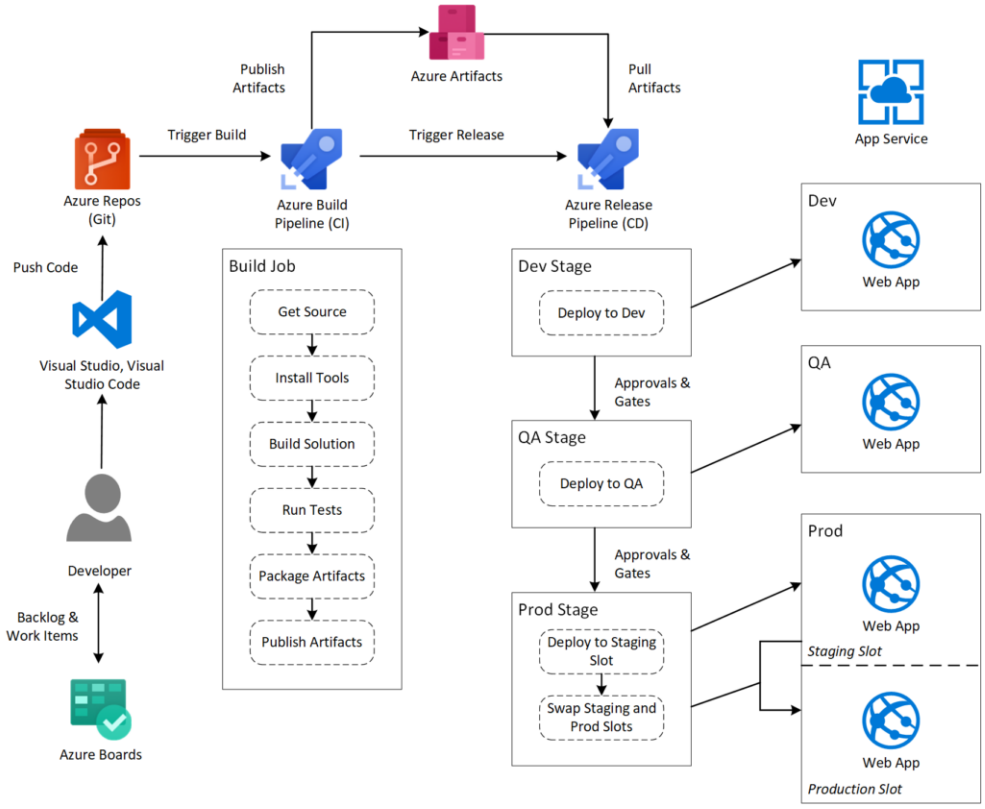

Figure 1: Azure DevOps CI-CD workflow

After CD pipelines are executed the application is continuously deployed in Azure portal. Figure-2 represents a CD-CD pipeline which generates a containerized docker image of the application, this image can be configured to automatically stored in a public registry like DockerHub or Azures own private registry Azure container registry. From where other deployment platform like Azure App service pan, Azure Kubernetes Service (AKS), Azure container instance (ACI) pulls the image through web hooks and deploys the application. All of these steps can be defined by using the azure pipelines service, which will automate the entire system.

1.6 Advantages

1. Since Azure DevOps is a Software-as-a-service(SaaS) product it has many advantages like :[19]

   - No manual infrastructure maintenance is needed.
   - It is much more intuitively clear and user-friendly in comparison to similar platforms.
   - Users don't need to worry about upgrading or patching up the tool-chain, thus massive organizations using the DevOps service will not experience any slowdown in CI-CD pipelines.

2. Categorized Built in Tasks — In Azure DevOps tasks are categorized based on the nature of operation. For Instance, Build tasks,Utility tasks,Deploy tasks etc. This

[19] `https://www.softacom.com/en_azure_devops`

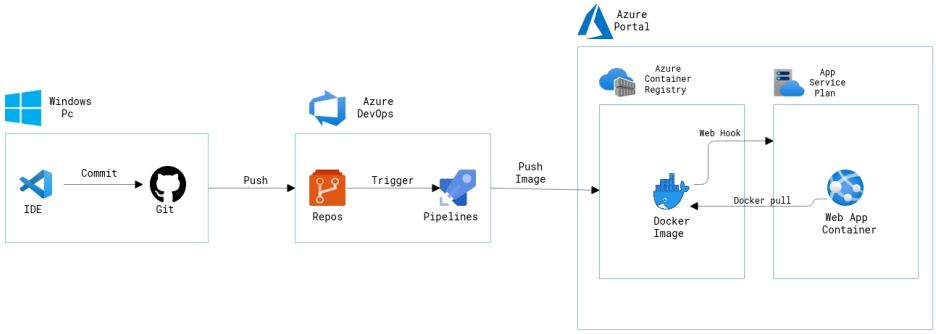

Figure 2: Azure DevOps CD to Portal

facilitates user in adding desired tasks to their pipeline in an organized manner.[20]

3. Group Tasks — It provides the option to encapsulate a sequence of tasks, already defined in a pipeline, into a single reusable task ,just like any other task.

4. It is platform agnostic. Which means, Azure DevOps is designed to run on any platform (Linux, macOS, and Windows) or language (e.g., Android, C/C++, Node.js, Python, Java, PHP, Ruby, .Net, and iOS apps).

5. It is cloud agnostic. That means, Azure DevOps can work in conjunction with AWS and GCP or even with an on-premise server.

1.7 Disadvantages

1. Azure Pipeline workflow is straightforward. It can't if-else or switch-case constructions. This makes it more difficult to develop complex workflows.

2. If the project requires exploratory testing such as alpha,beta testing then the cost structure becomes expensive.

3. Documentations of azure are not always up-to-date and sometimes missing.

4. Integration with non Microsoft tools sometimes can be problematic.

1.8 Price Plans

Azure DevOps offers two different plans for purchase :

- Basic Plan : It is free for up-to 5 users, from 6 users on-wards the organization will be charged 6 USD per user every month.
- Basic + Test Plans : It costs 52 USD per user per month.

[20] `https://ymedialabs.medium.com/the-pros-and-cons-of-jenkins-vs-azure-devops-469c66140b4d`

Part IV

AWS

1 CI-CD on AWS

1.1 Description

Amazon Web Services is a subsidiary of Amazon that offers a metered pay-as-you-go cloud computing platforms and APIs to individuals, businesses, and governments. These web services for cloud computing include a range of simple abstract technical infrastructure and distributed computing building blocks and tools.

1.2 Services needed for CI/CD

AWS offers more than 200 products and services, including computing, storage, database, networking, application services, deployment, management, analytics, machine learning, developer tools, and IoT tools. Among these, the most popular are Amazon Elastic Compute Cloud **(EC2)**, Amazon Simple Storage Service **(Amazon S3)**, Amazon Elastic Block Store **(EBS)**, Amazon Connect, and AWS Lambda. Down below, a brief description of the AWS services related to CI/CD is given[21].

1.2.1 Amazon Elastic Computing Cloud (EC2)

EC2 enables users to have a virtual cluster of computers accessible at any time via the Internet. It provides the most comprehensive computing platform, with various options for processor, storage, networking, operating system, and purchase model. EC2 facilitates flexible application deployment by offering a web service that allows users to boot an Amazon Machine Image (AMI) and customize the virtual machine, which Amazon refers to as an "instance," with the software they prefer. The operating systems currently available to use with Amazon EC2 instances include: Amazon Linux, Windows Server 2012, CentOS 6.5 and Debian 7.4.

1.2.2 AWS CodeCommit

AWS CodeCommit is a fully managed source control service that hosts secure Git-based repositories. It makes it easy for teams to collaborate on code in a secure and highly scalable ecosystem. CodeCommit eliminates the need to operate our own source control system or worry about scaling its infrastructure. One can use CodeCommit to securely store anything from source code to binaries, and it works seamlessly with the existing Git tools.

[21] https://aws.amazon.com/products/developer-tools/?nc2=h_ql_prod_dt

1.2.3 Amazon Simple Storage Service (Amazon S3)

Amazon Simple Storage Service is an object storage service that offers industry-leading scalability, data availability, security, and performance.

1.2.4 AWS CodeBuild

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. CodeBuild provisions, manages, and scales our build servers continuously and processes multiple builds concurrently, so the builds are not left waiting in a queue.

1.2.5 AWS CodeDeploy

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of computing services such as Amazon EC2, AWS Fargate, AWS Lambda, and on-premises servers. AWS CodeDeploy makes it easier to rapidly release new features, helps avoid downtime during application deployment, and handles the complexity of updating applications.

1.2.6 AWS CodePipeline

AWS CodePipeline is a fully managed continuous delivery service that helps automate the release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of the release process every time there is a code change, based on the release model we define. We can easily integrate AWS CodePipeline with third-party services such as GitHub or with our own custom plugin.

1.2.7 AWS CloudFormation

AWS CloudFormation gives us an easy way to model a collection of related AWS and third-party resources, provision them quickly and consistently, and manage them throughout their lifecycles, by treating infrastructure as code. A CloudFormation template describes our desired resources and their dependencies so we can launch and configure them together as a stack. We can use a template to create, update, and delete an entire stack as a single unit, as often as we need to, instead of managing resources individually.

1.3 CI-CD Process

The following example uses two separate AWS accounts for development and production. In summary, the example has the following workflow[22]:

[22]https://aws.amazon.com/blogs/devops/complete-ci-cd-with-aws-codecommit-aws-codebuild-aws-codedeploy-and-aws-codepipeline/

1. A change or commit to the code in the CodeCommit application repository triggers CodePipeline with the help of a CloudWatch event.
2. The pipeline downloads the code from the CodeCommit repository, initiates the Build and Test action using CodeBuild, and securely saves the built artifact on the S3 bucket.
3. If the preceding step is successful, the pipeline triggers the Deploy in Dev action using CodeDeploy and deploys the app in dev account.
4. If successful, the pipeline triggers the Deploy in Prod action using CodeDeploy and deploys the app in the prod account.

The following diagram illustrates the workflow:

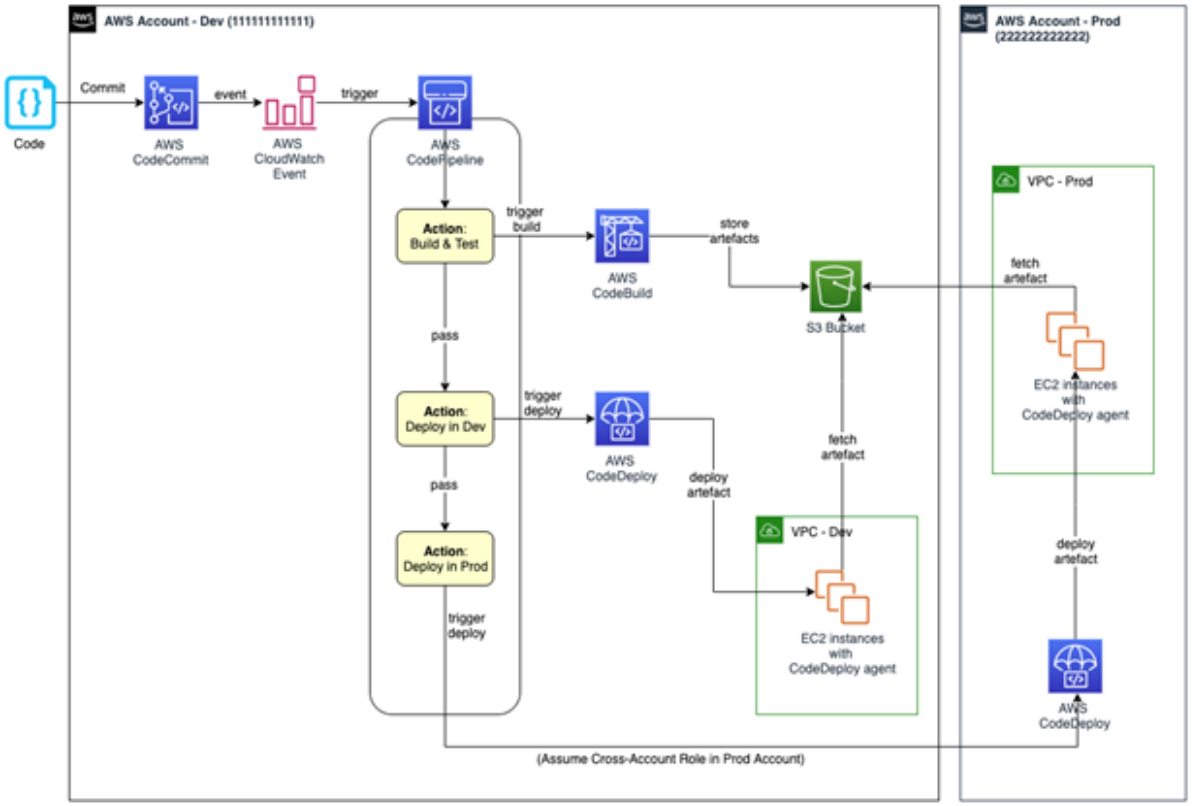

Figure 3: Workflow of CI/CD pipeline in AWS.

1.4 Estimated Cost

The total cost of running a CI/CD pipeline on AWS depends on the AWS services used in your pipeline. Monthly charges will vary on your configuration and usage of each product, but if we use all the AWS services mentioned above and accept the default configurations, we can expect to be billed around $15 per month.

1.5 Advantages

Companies and individuals prefer AWS as their cloud provider because of the numerous AWS benefits it provides.

- **User-friendly:** AWS is easy to use as the platform is specially designed for quick and secure access.There won't be any problem for a new applicant as well as for an existing applicant.
- **Cost-effective:** AWS offers a pay-as-you-go pricing method, which means that a company will only pay for the services that it needs and has used for a period of time.
- **Scalable and Elastic:** AWS is scalable because the AWS Auto Scaling service automatically increases the capacity of constrained resources as per requirements so that the application is always available. If you use fewer resources and you don't need the rest of them, then AWS itself shrinks the resources to fit your requirement.
- **Scope of operations:** AWS has a huge and growing array of available services, as well as the most comprehensive network of worldwide data centers.

The Gartner report summed it up, saying, "AWS is the most mature, enterprise-ready provider, with the deepest capabilities for governing a large number of users and resources."

1.6 Disadvantages

These are the limitations of Amazon Web Services:

- **Complex Price Plans:** Amazon's big weakness relates to cost. While AWS regularly lowers its prices, many enterprises find it difficult to understand the company's cost structure and to manage those costs effectively when running a high volume of workloads on the service[23].
- **Limitations OF Amazon EC2:** AWS sets default limits on resources which vary from region to region. These resources consist of images, volumes, and snapshots. You can launch the limited number of instance per area. It also provides limited information for the resources managed by Amazon EC2and Amazon VPC console[24].
- **Technical Support Fee:** AWS charges you for immediate support and you can opt for any packages among 3 which are- **Developer**, **Business**, **Enterprise**
- **Complex Organisation:** AWS can be overwhelming with the vast amount of services they provide. It can be tough to pick the right option to fit the goal you're trying to accomplish

[23]https://www.datamation.com/cloud/aws-vs-azure-vs-google-cloud/
[24]https://data-flair.training/blogs/aws-advantages/

Part V

Google Cloud

1 CI-CD on Google Cloud

1.1 Introduction

Google Cloud Platform is one of the most used public cloud platforms and is still continuing to grow. It offers a cloud services that run on same infrastructure that Google hosts their product such as Google search, Gmail and Youtube. After Launching of Google App Engine, we have seen other services introduced and the list is continuing to rise.

1.2 Setting up a CI/CD pipeline for data-processing workflow

Setting up a continuous integration/continuous deployment is done by implementing CI/CD methods with manage products on Google cloud. It ensures high quality,maintainability and adaptability of data process and workflows.The methods are as follows:

- Version Control of source code.
- Automatic building,testing and deployment of apps.
- Environment isolation and separation from production.
- Replicable procedures for environment setup.

1.3 Deployment Architecture

- **Cloud Build** to create a CI/CD pipeline for building, deploying, and testing a data-processing workflow,as well as the data processing itself. It is a managed service that uses Google Cloud to run the build. A build is a collection of build steps, each of which is run in its own Docker container.
- **Cloud Composer** is used to define and run workflow steps such as data processing, testing, and verification. It is a managed Apache Airflow service that allows you to create, schedule, monitor, and manage complex workflows.
- **Dataflow** is the serverless execution service from GCP for data-processing pipelines written using Apache Beam. Apache Beam is an open-source, unified model for defining both batch and streaming data-parallel processing pipelines.

1.3.1 The CI/CD pipeline

At a high level, the CI/CD pipeline consists of following steps:

- Cloud Build packages the sample application into a JAR file using the Maven Builder. The Maven Builder is a container with Maven installed in it.Maven runs the task When a build step is configured to use the Maven Builder
- Cloud Build uploads the JAR file to cloud storage
- Cloud Build runs unit tests on data-processing workflow code and deploys the workflow code to the Cloud Composer.
- Cloud Composer picks up the JAR file and runs the data-processing job on Dataflow.

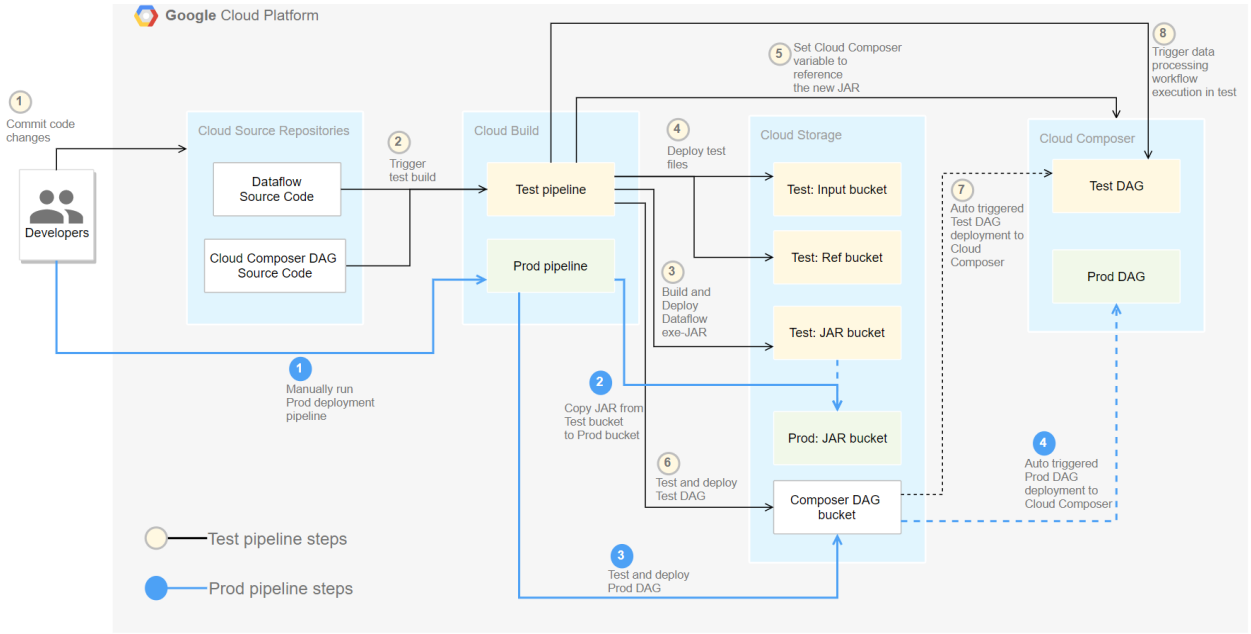

Figure 4: Workflow of CI/CD pipeline in GCP.

We can see in the figure 4, the deployments to the test and production environments are separated into two different cloud Build pipelines- a test and a production pipeline.[25]
In the preceding diagram, the test pipeline consists of the following steps:

1. A developer commits code changes to the Cloud Source Repositories.
2. Code changes trigger a test build in cloud build.
3. Cloud Build builds the self-executing JAR file and deploys it to the test JAR bucket on cloud storage.
4. Cloud Build deploys the test files to the test-file buckets on cloud storage.

[25] https://cloud.google.com/architecture/cicd-pipeline-for-data-processing#the_cicd_pipeline

5. Cloud Build sets the variable in Cloud Composer to reference the newly deployed JAR file.
6. Cloud Build tests the the data processing workflow Directed Acyclic Graph(DAG) and deploys it to the Cloud Composer bucket on Cloud Storage.
7. The workflow DAG file is deployed to Cloud Composer
8. Cloud Build triggers the newly deployed data-processing workflow to run.

In the preceding diagram, the production pipeline consists of the following steps:

1. A Developer manually runs the production deployment pipeline in Cloud Build.
2. Cloud Build copies the latest self-executing JAR file from the test JAR bucket to the production JAR bucket on cloud storage.
3. Cloud Build tests the production data processing workflow DAG and deploys it to the Cloud Composer bucket on Cloud Storage.
4. The production workflow DAG file is deployed to Cloud Composer.

1.4 Pricing

Google Cloud Platform has no up-front costs, pay-as-you-go services, and no fees for termination. In addition, GCP stands out for for its discounts.

1.5 Advantages

GCP probably benefits from the simple fact that it is owned by google. Apart from that, there are some benefits of using GCP.

- Google Cloud Platform(GCP) is designed for cloud-native businesses
- Commitment to open source and portability
- Deep discounts and flexible contracts
- DevOps expertise

1.6 Disadvantages

GCP has some disadvantages like other platform.

- GCP is still continuing to grow.
- It has less feature and service than AWS and Microsoft Azure.

Part VI

Cloud Tools Comparison

1 Particular tool Selection through comparison

The CI/CD process of most software projects can be broken down into these parts:

- Version control
- Code Review
- Building
- Automated Testing
- Deployment

Hence, these points will be the dimensions of the tool-chain comparison.

<table>
<tr><th></th><th>Azure</th><th>Google Cloud</th><th>AWS</th></tr>
<tr><td>Version Control</td><td rowspan="5">DevOps/GitHub Enterprise</td><td>Cloud Source Repositories</td><td rowspan="2">CodeCommit (CodeStar)</td></tr>
<tr><td>Code Review</td><td>–</td></tr>
<tr><td>Build</td><td rowspan="3">Cloud Build</td><td>CodeBuild / CodePipeline (CodeStar)</td></tr>
<tr><td>Test</td><td>CodeBuild / CodePipeline (CodeStar)</td></tr>
<tr><td>Deploy</td><td>CodeDeploy / CodePipeline (CodeStar)</td></tr>
</table>

Figure 5: Tool-Chain Comparison

After analysing Figure-5, following set of deductive comparison could be made:[26]

- **Service Comparison** - Azure DevOps offers a complete service suite, a set of services that integrates seamlessly. There are sub-parts but everything is coherent and there is one tool for one job. Google is half-way there with a conjunction of two services. Whereas, AWS has more of a toolbox. It has a myriad of services that a user needs to find and combine in order to establish a secure pipeline. AWS presents a steeper learning curve to the new-comers, which can slow down the entire development process.

[26] https://www.inovex.de/en/cloud-native-cicd-part-2-azure-devops-vs-aws-vs-google-cloud/

- **Version Control and Code Review** - Azure DevOps Repos workflow is exactly as same a GitHub Enterprise, which facilitates better code review capabilities. AWS CodeCommit supports pull request workflows like GitHub or GitLab but it's just not as user-friendly and intuitive as them. Google Cloud Source Repositories are a remote git back-end in the most basic sense. It's possible to push and pull from it. Not more, not less. Code reviews on feature branches are rather important and not possible with GCP Cloud Source Repositories.

- **Build and Test** - For building and performing automated tests, Azure and AWS provides a myriad of options through their management console. It is possible to build an application regardless of its development language and integrated build tools like Gradle, Maven or Ant. However, in Google Cloud Build the service comes with a limited set of around 20 official cloud builder Docker images. These official images support most common tool chains. However, on top of those official images, there are community images for a lot more project (Terraform, Ansible, Android etc.) . GCP does not provide pre-built containers for those images. A developer has to clone the needed spec, build the image and push it to the container registry of the project before usage.

- **Deployment** - In terms of deploying an application, all of the cloud platforms provide a range of options from server-less options to standard containerized web apps and more. However, Microsoft Azure trumps AWS and GCP in this category for one reason. Since Azure DevOps is a independent service suite, it is not limited to deploying applications to in-house solutions like Azure Portal. Azure DevOps can deploy an application in other cloud vendor portals like AWS Managemnt Console or GCP Platform or even to on-premise servers.

After analyzing all the core parts of the CD-CD process of all three cloud providers, Microsoft Azure DevOps seems the most viable option to implement a CI-CD pipeline in the cloud for the following reasons:

- Azure DevOps Repos are essentially the same as Github and GitHub Enterprise, since Microsoft acquired GitHub in 2018. Similar architecture and workflow as GitHub means a familiar environment for the developers which provides convenience and also can speed up development process.

- It provides IaaC support for almost all the popular tools such as Chef, Puppet, Ansible and Terraform through official images for configuring and provisioning development, QA and production environments. In other platforms like GCP, third-party images are needed to use those tools.

- There are less possibility of vendor lock-in in Azure DevOps. Because it can deploy applications to multiple cloud and on-premise solutions. Thus, if the need of vendor migration arises the process will be seamless.

- Azure DevOps service is completely free for up-to 5 users. From 6 users and onwards a small fee of 6 USD is charged on a bi-monthly basis per user. Unlike other cloud services like AWS and GCP, there are no cost for pipeline execution or storing application code in Repos. Application artifact storage is also free up-to 2 gigabyte.

Part VII

Proposed Methodology

1 Proposed methodology for CI-CD on cloud automate release

In this section the important components configuration mechanism will be discussed for CI-CD implementation.

1.1 Version Control

At first the version control is needed to organize the code changes, configuration scripts and documetations. For this **Git** will be our version control team. It is very well established in software release processes now-a-days. Popualar version control tools like **GitHub, GitLab, Azure Repos Git or Bitbucket Cloud** can be used in this case almost all the CI-CD on cloud tool gives the facilation of these tools as well.

1.2 Build Tool

Another important tool is to build the application in CI-CD process. This tool will help the project to create artificates and resolve dependency by it's own. One of these **Gradle, Maven** two build tool will be used in our CI-CD implemantation.

1.3 Deployment Target

Azure Pipelines can be used to deploy project code to multiple targets. such as, container registries, virtual machines, Azure services, or any on-premises or cloud target. The publised project code is needed to be deployed in a manner so that it can be delivered to he clients with minimal effort. That's why docker images is used greatly at the present. And for this we can use service like **Azure DevOps, CodeDeploy and Cloud Build** to create deployment target. Azure Devops usage will help the project development team to deploy it to different services like aws management console and google cloud platform as well as its own service, Azure Portal.

1.4 Configuration Scripts

For CI-CD implementation one of the main part is configuration scripts. For different services setup this scripts are used to handle the automation process. For example, if Azure Devops is used as a CI-CD tool than following scripts need to be configured.

- To define a workflow that describes how test, build, and deployment steps are run.
- The seperations of concerns is maintained such as Dev, Staging and Prod is maintained on the sripts.
- The scripts are mainly consists of different tasks and every task is responsible for specific job execution.
- This scripts will be controlled by some kind of agents or agent pool to maintain order in between the tasks.
- On the sripts differernt environment will be defined to execute the scripts on correct environment.
- In the end, the scripts will be run on the tool to execute every described task.
- Additionally trigger is also used to tell the pipeline when to run the scripts.

This scripts are created and kept on the version control to use further on the development.

1.5 Testing

It is one of the main part for CI-CD automation. expecially for continuous integration. This part will determine the code changes are deliverable or not. That's why unit testing, Integration testing, acceptance testing and regresion testing are integrated in the CI-CD automate release process. To automate testing the following concepts need to be used in the project.

- **Test Frmaework**: Test framwork is to organise the test cases, test plan and coverages. **TestNG** is widely used for java based application. Test Framework will provide a detailed report after each execution to give an overview for test execution.
- **Test Automation tool**: To automatically run the application based on the test case automate testing tool is needed. For web based application **Selenium** is widely populer due it's cross browser platform facilation.
- **Reporting Tool**: Some kind of reporting tool can also be used in the testing automation. This tool will generate a formatized report and more presentable to the user. Some popular reporting tools are **ReportNG, Extent Report**. Additionally, this is an optional part since test framwork can handle it by itself.

This testing will need to be integrated in the CI-CD as a idependent jobs or part of their process as well. In most cases Unit Testing are integrated as part of CI-CD to capture big issue at the begining with less execution time.

1.6 Infrastructure as Code

Infrastructure as Code or IaC can make a CI-CD pipeline faster and more consistent by automatically configuring and provisioning resources in a cloud environment. Cloud specific tools such as AWS CloudFormation or Azure Resource Manager(ARM) can be used to provision resources in their respective cloud platforms through templates that are generally JSON or YAML files. However, in order to be cloud agnostic third party tools like Chef, Pupet, Ansible or Terraform should be preferred. Azure and AWS offers the option to integrate IaC templates from third party providers into their continuous delivery pipelines. Scripts from third party tools such as Cookbooks of Chef, Manifests of Puppet, Playbooks of Ansible or HCL of Terraform can be used to dynamically provision and manage resources and configure servers in the cloud.

1.7 Security

Security is one of the important matter for CI-CD on cloud. All the cloud tools naturally provide security by its own. Especially User-Role Management, credential management as security concern.

- **User-Role Management**: This specific parts of security is responsible for access related issues primarily. Not all the team members will not have all the access of the CI-CD process. The neccessity is defined beforehand and roles are assigned to the user as per that decision. In this way security breach can be tracked and amintained much more smoothly.
- **Credential Management**: Credential of users Will be managed by the CI-CD tools. They should make some process to secured this access by access token or any other security management mechanism. Credentials for creating and running a pipeline job should never store in a local machine and also on the third party tools. It should be encripted and saved in additional password management tools.

Security is a main concern because in CI-CD we are using different tools for IaC, testing which can be used to breach the software more easily and frequently.

Part VIII

Conclusion

1 Conclusion

CI-CD on cloud tool is used for the betterment of the development, testing and deployment process. The main concern of this kind of tools is to reduce the overhead and stress from the team members. The cloud application is also making the release process more inclined through automation steps. Software can be released with minimal effort. Risk of introducing bugs are decreased by a big margin. It Speeds up the release steps of a software as well. Rollback to the previous stable software version in case of any failure is also only a click way from the teams. There is no need to notify user or waste their precious times to deploy the product on their environment and make necessary changes in the environment as well. CI-CD is still rather a new topic in the software development culture but it will be recognised and practised by more in the coming days.