Week 2. Git & Cloud Architecture

제1강. Intro to Version Control

(1) What is the version control?

- The version control is a system managing and tracking code changes over time.

- It can help the retrieval of earlier versions, track who modify codes and manage modifications made by multiple people.

- It is not just a tool but a fundamental aspect of modern software development.

(2) What is the role of version control in software development.

- The role of version control in software development is to track changes.

- Understanding when and why the changes occurred and by whom to assist in debugging and understanding the evolution of projects.

- Another role of version control is the collaboration, enabling multiple people to work on the same project simultaneously without overwriting each other's work.

(3) There are 3 concepts to a version control system.

1. Repository

- It is the database storing project files and revisions. This is where code for infrastructure, pipelines and AWS are stored.

2. Commit

- It is a snapshot of each file at a particular point in time, along with a message describing your changes.

3. Branch

- It is the parallel version of repository, created to develop features, fix bugs and safely experiment new codes and ideas.

(4) 3 types of version control system.

- There are local, centralized and distributed version control systems.

1. Local

- In a local version control system, all versioning activities are performed on a single machine. For example, storing file copies in different directories on the same computer. This approach is simple but can be cumbersome and lacks collaboration features.

Copies here versions?

2. Centralized

- In a centralized version control system (CVCS), old versions of files are stored in a single central server repository and clients can update and commit files from a central place, suitable for the collaboration with teammates, as all project files and version histories are kept in one place, such as Subversion.

3. Distributed

- In a distributed version control system (DVCS), there is main server repository, but each user also has a complete copy of the central repository on their own computer. Users can push and pull files to and from the main server, and they can also commit changes locally on their machine before sharing them with others.

It offers more options to collaboration by merging changes made on their own as needed and provides redundancy by allowing for offline work and a full copy of the central repository, such as Git, which is the most popular version control system.

(5) 3 key best practices in version control

1. Writing commit messages

- It is clear and descriptive of what was changed and why.

2. Committing changes to our branch frequently

- It is better to commit changes frequently instead commit large and complicated changes just in one go.

3. Branching strategy

- We need to use branches for feature development, bug fixes and experimentation.

제2강. Intro to Git

- We are going to learn about what git is and common git commands.

(1) What is git?

- Git is a distributed version control system, designed to handle any project of any size with speed and efficiency.

- Git allows any developer to check the entire history of codebase, enabling autonomous work and redundancy.

- Git facilitates powerful branching and merging, crucial for modern software development workflows.

(2) Key Git Commands

1. git init

- You can install or initialize a local storage location to track all version-controlled files on your computer for collaboration.

- This command will make a local Git repository on your computer.

2. git clone url

- This command is used to copy the entire remote repository to your computer, and the URL specifies the exact remote location from which users want to copy the repository.

- Remote repositories for copying are often hosted on GitHub accounts.

(3) Staging and committing changes

1. git status

- Check the status of changes

2. git add [file]

- Stage a file for commit.

- You change and modify code in your working area and then move it to the stage area for commit.

3. git commit -m “commit message”

- Commit staged changes with a descriptive message

- Once all the desired changes are staged, you apply all the changes to your repository using the commit command.

4. Branching and merging

1. git branch

- List, create or delete branches

- When you create a new Git repository, a main (often called as main or master) branch is also created where all you commits are initially reflected.

2. git branch [branch-name]

- Create a new branch

- By creating a new branch separate from the main branch, you can fix bugs or create new features without affecting the main branch.

3. git checkout [branch-name]

- Switch to a different branch.

- When you switch to another branch, you can update files in that branch without affecting the branch you were previously on. These changes will only be reflected in the branch you switched from if you merge or commit them later.

4. git merge [branch]

- Merge a specified branch into another branch usually main or major branch

- Any changes made in the branch being merged will be incorporated into the current branch you are on, such as the main branch, thus reflecting those updates in the main project.

(5) To view any of our git changes

1. git log

- View commit history.

- List of all the commits that have been made to the repository.

2. git diff

- Show changes between commits, branches, etc

- It can show the differences between your working directory and the last commit or between two different commits.

(6) To undo any changes

1. git revert [commit]

- Create a new commit that undoes changes from a specific commit and reverses the changes made by a previous commit.

2. git reset

- Reset the current's HEAD to a specified state.

- Reset the HEAD, staging area, and the working directory to a specified state, effectively discarding the entire changes after the specified commit.

(7) Woking with Remote Repositories

1. git push [remote-name] [branch-name]

- Push local branch to a remote repository

- Share your changes with others or update the remote repository after working on your project locally.

2. git pull [remote-name] [branch-name]

- Fetch and merge the latest changes from a remote branch into your current branch, ensuring that your current branch is up-to-date with the remote branch.

제3강. Understanding GitHub

- Enter github web page https://github.com/

- Github is a web-based platform for version control and collaboration. It uses Git, a distributed version control system to track changes in source code throughout the software development process.

- Github is developed for developers and engineers to collaborate on projects and is the biggest platform for hosting open source and private projects.

- After setting up a private user account, we create an SSH key to allow secure access to our Github repository, enabling us to make changes and push code without needing to enter username and password repeatedly.

- Creating SSH key for cmd in Widows **(관리자 권한으로 실행)**

1. Generate an SSH key

ssh-keygen -t rsa -b 4096 -C jiseo@ucdavis.edu

2. Press Enter until it shows you “Key fingerprint and Key’s randomart image”

3. Copy the public key to the clipboard:

clip < C:\Users\sjh50\.ssh\id_rsa.pub

4. Add the SSH Key to your Github Account:

(1) Log into **Github.**

(2) Navigate to Settings > SSH and GPG Keys > New SSH key.

(3) Paste the SSH key from your clipboard into the “Key” field.

(4) Give your key a descriptive title and click Add SSH key.

5. Test the SSH connection:

Run the following command in your terminal: ssh -T git@github.com

6. You should see a message like this:

Hi AWSwithJihyun! You've successfully authenticated, but GitHub does not provide shell access.

7. If not, use the following command in your Command Prompt to view your public key by typing C:\Users\sjh50\.ssh\id_rsa.pub and copy and paste the entire public key into the “Key” field in Github and follow the same steps from Step 4 to Step 6.

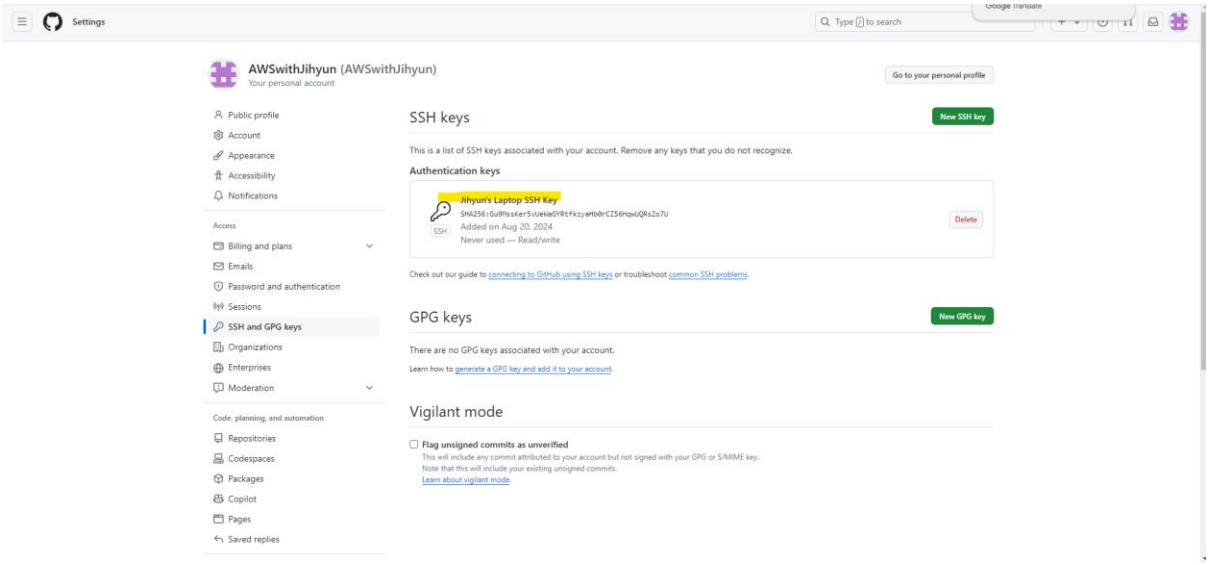
The following SSH key was added to your account:

Jihyun's Laptop SSH Key

SHA256:Gu9MssKer5vUeWaGYRtfkzyaHb0rCZ56HqwUQRsZo7U

If you believe this key was added in error, you can remove the key and disable access at the following location:

https://github.com/settings/keys



제4강. Creating a new repository

1. Create a new public repository in your dashboard page.

2. After creating the public repository, you can clone this repository into your local machine and setup it under the created SSH key to store README.md file into this repository.

3. In the terminal of Git bash, you can store this repository in your 'Documents' while creating a new folder under the 'Documents'.

% cd Documents (cd means going into the 'Documents' folder in your local machine under the path ('/c/Users/sjh50')

$ cd ~/Documents (cd changes the directory into the 'Documents' folder)

% mkdir cloud-bootcamp (which means that make a new folder (directory) using the terminal under the 'Documents' and call this new folder as 'cloud-bootcamp')

$ mkdir cloud-bootcamp

% cd cloud-bootcamp (cd means going into the folder here is 'cloud-bootcamp' and change the directory to the 'cloud-bootcamp' file.)

$ cd cloud-bootcamp

cloud-bootcamp % pwd (pwd enables us to see the full path, confirming where you are.)

$ pwd

cloud-bootcamp % echo "# cloud-bootcamp" >> README.md ('echo' stands for printing new text into the file, which means that print '# cloud-bootcamp' text into the README.md file directly.)

echo "# cloud-bootcamp" > README.md

cloud-bootcamp % cat README.md (you can see '#cloud-bootcamp' is written on the file)

cat README.md (cat displays the contents of the file "README.md". You can see the "# cloud-bootcamp" in the file.)

cloud-bootcamp % git init (which means that we can initialize a new repository inside the current folder we have already created, making a local repository in the local machine.)

git init (we can make an empty Git repository in C:/Users/sjh50/Documents/cloud-bootcamp/.git/)

cloud-bootcamp % git status (you can find README.md file is red and this file is untracked, indicating that it has to be pushed and committed to the public repository. git status finds out which files are staged for the next commit in our local repository (green) and which files are not tracked (red) needed to be staged for the next commit.)

git status

cloud-bootcamp % git add . (We make the README.md file as tracked by using 'add' command to commit and push the README.md file into the public repository. "." Stands for all the files in the current directory needed to be staged for the next commit.)

git add .

cloud-bootcamp % git status (from which you can see the file, README.md is green indicating that these files are ready to push and commit into the public Github repository.)

git status

cloud-bootcamp % git commit -m "first commit readme" (from which we can commit the file, README.md and '-m' flag allows you to include a commit message, crucial for tracking changes.)

git commit -m "first commit readme"

cloud-bootcamp % git status (There is nothing to commit because we have already pushed and committed the README.md file into the public repository)

git status (You can see "On branch master, nothing to commit, working tree clean)

cloud-bootcamp % git branch -M main

git branch -M main (It is now a common convention to rename 'major' into 'main' for the primary branch in Git repositories)

cloud-bootcamp % git remote add origin git@github.com:AWSwithJihyun/cloud-bootcamp.git

git remote add origin git@github.com:AWSwithJihyun/cloud-bootcamp.git

(This command adds a remote repository link named 'origin' to your local Git repository, pointing to the public remote Github repository at the provided URL.)

cloud-bootcamp % git push -u origin main (after that we can enter github.com/AWSwithJihyun/cloud-bootcamp, and then we can see the README.md file is pushed and committed to our repository and 'cloud-bootcamp' is written newly on that file.)

git push -u origin main

(This command pushes your local commits to the remote repository on Github. The '-u' flag sets 'origin main' as the default push/pull location for future commands.)

cloud-bootcamp % open -e README.md (open the README.md file for editing and then we input 'Updating file' and just save the file.)

notepad README.md *using notepad on Window for editing the file, writing down “Updating file” and save the change.

cloud-bootcamp % git status (from which we can see the README.md file is red and modified.)

git status

cloud-bootcamp % git add.

git add .

Cloud-bootcamp % git commit -m “updated readme” (just write down the commit message between “”)

git commit -m “updated readme”

git push

After that we refresh the URL (github.com/AWSwithJenny/cloud-bootcamp/blob/main/README.md)

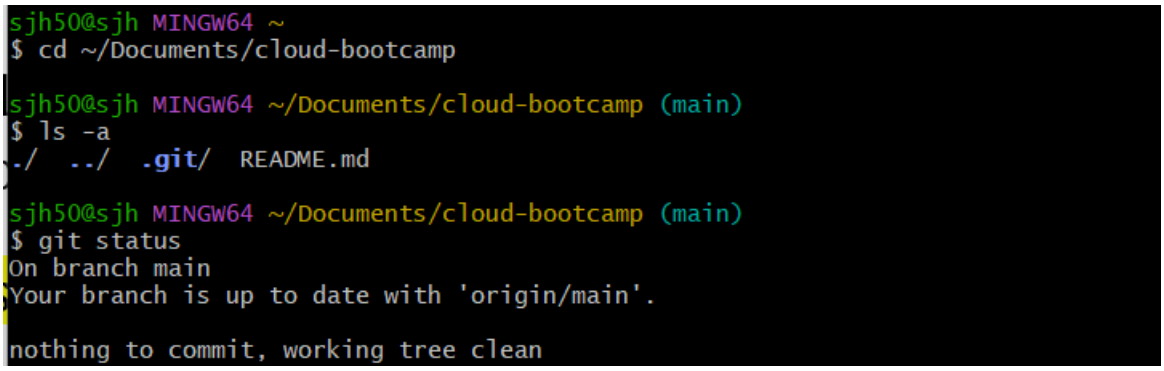
And we can see “Updating file” text was uploaded.

제5강. Git Workflow

Firstly, before trying to update the README.md file, we need to make sure that we are in the right local directory.

- cd ~/Documents/cloud-bootcamp is to check whether you are in the right Git repository such as (~/Documents/cloud-bootcamp).

- **ls -a** and inside the cloud-bootcamp directory, there should be a .git repository, which contains all the information about your Git repository.



- git status is an important command to check what changes are being tracked.

- open -e READE.md and the file will open and write down “Making new changes git commands” and save the file and go back to the terminal

- git status and run git status again and we can see these changes are not staged for commit (modified: README.md is on red) and we need to stage this file to be committed.

- git add . OR git add README.md

: When we use . in the command, we can push all the files in our repository to the stage but we can just designate the file name that we want to commit.

- git status and we can enter the git status again whether this file is added. We can see the file name is on green, indicating that this file is ready to be committed.

- git commit -m ‘updated Readme file for tutorial’ and this command commits this file to our Github and we need to comment what kinds of changes we have made by adding the commit message.

- git status and we can see we have committed this file and need to git push to publish this change to the remote repository.

- git push and then we can this change on the remote repository.

- When you work in any project team, you need to start with a branch.

- git branch and we can see that we are now in the main branch.

- git branch tutorial/git and this command means that we are still in the main branch but now we have created another branch called as ‘tutorial/git’ so now we need to go into this new branch because we do not want to make any direct change to the main branch.

```
soleyman@Soleymans-MacBook-Pro cloud-bootcamp % git branch
* main
  tutorial/git
```

- git checkout tutorial/git and this command let us switched to another branch ‘tutorial/git’.

- git branch and we can see that we are now in the ‘tutorial/git’ branch.

```
soleyman@Soleymans-MacBook-Pro cloud-bootcamp % git branch
  main
* tutorial/git
```

- open -e README.md and open the README file again and put “Making changes from tutorial/git” and save this file.

- git status

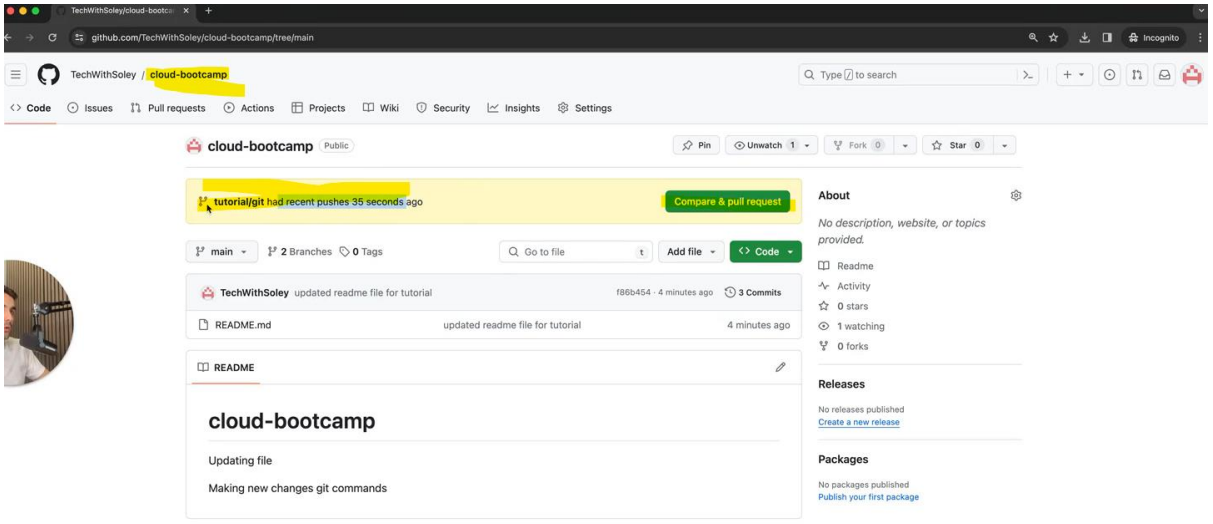
- git add .

- git commit -m ‘added to README from tutorial branch’

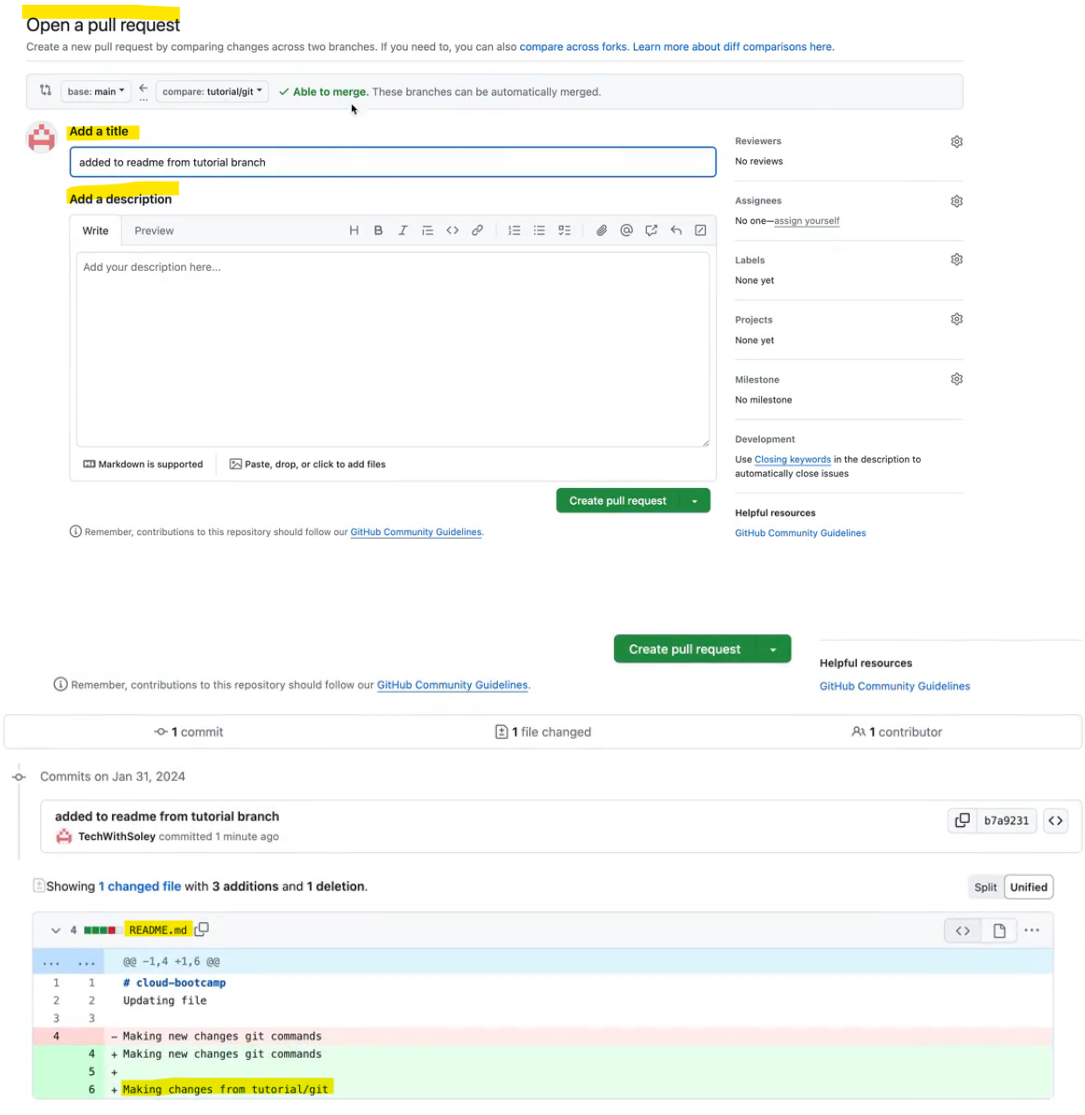
- git push but when I run this code, the terminal recommends using the below command

- git push –set-upstream origin tutorial/git

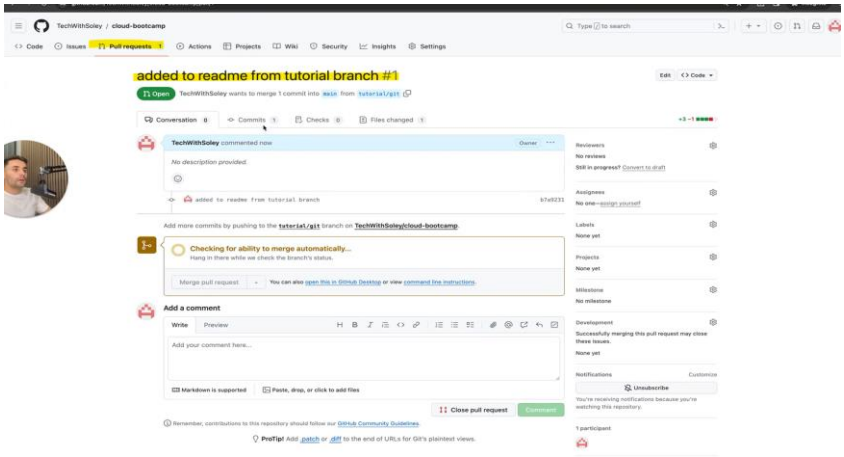
- git status and we can see we have made changes on ‘tutorial/git’ branch and we need to merge this change to the main branch on Github by clicking ‘Compare & pull request’



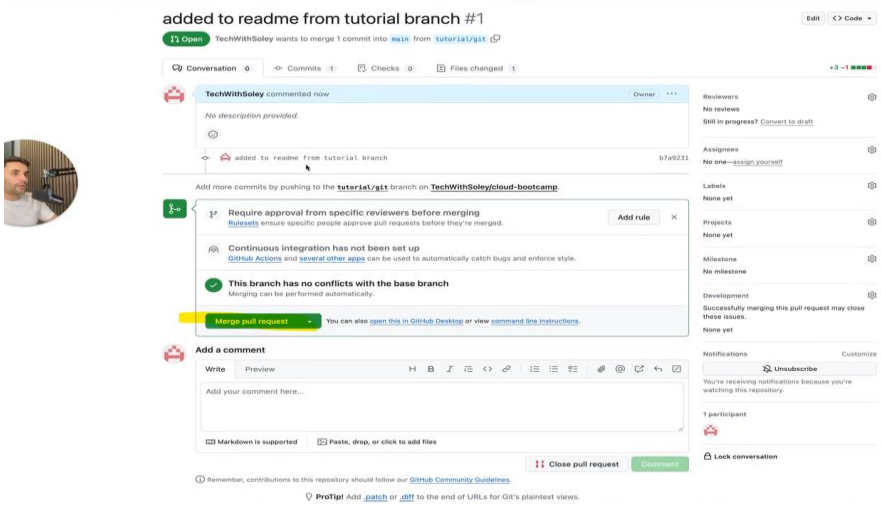
- While opening a pull request, we can see ‘add a title’, ‘add a description’ and what changes have been made in ‘tutorial/git’ branch.



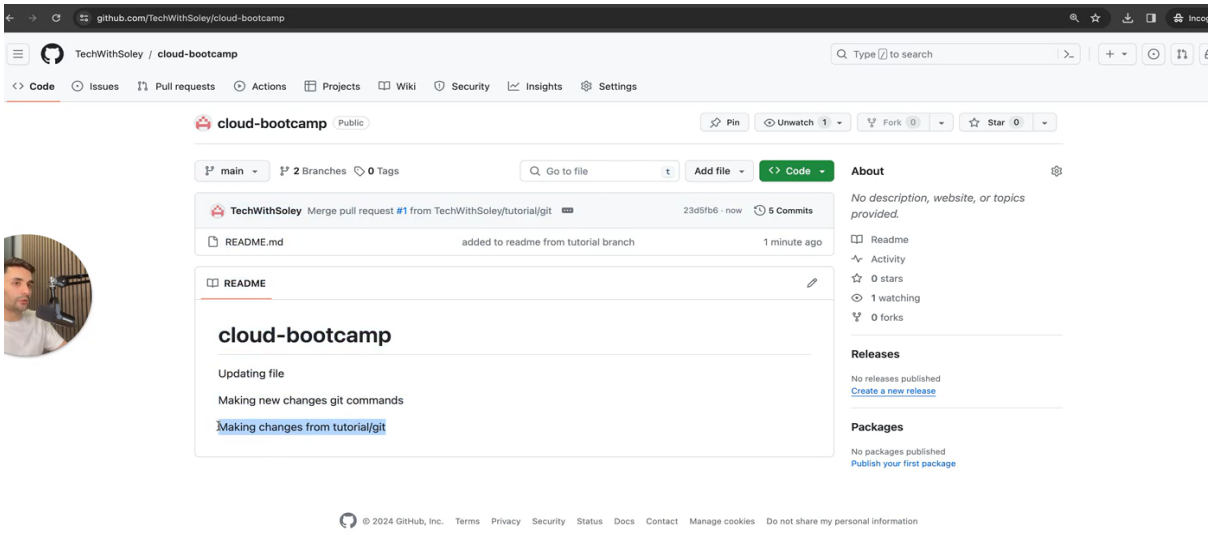
- Click ‘Create pull request’ and in the ‘Pull requests’ category, we can see these changes made in ‘tutorial/git’ branch.



- And now we have to merge that change to the main branch by clicking ‘Merge pull request’ and ‘confirm’. This process merged 1 commit into main from tutorial/git.



- We can see this change made to the main branch.



And back to the terminal to check the main branch and we can see that the main branch in our local repository is outdated, meaning that this change is not reflected in the main branch, even if we have pulled request to the remote repository in Github.

- git branch and we can see that we are still in the tutorial/git branch.

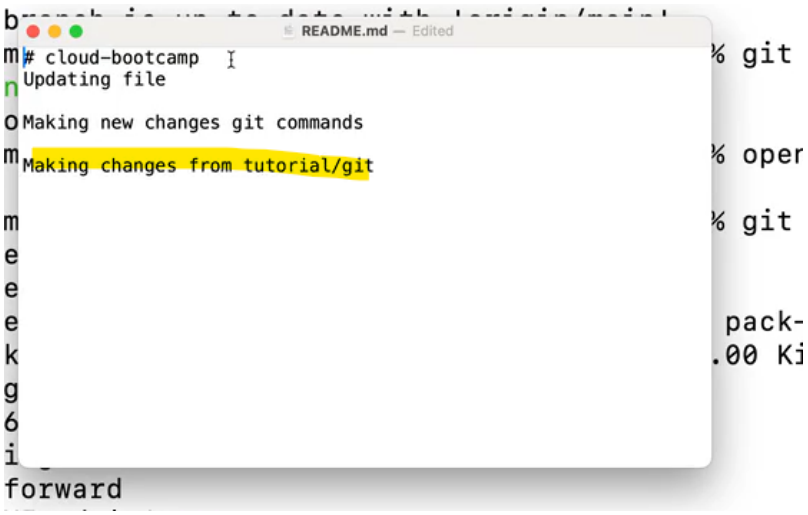
- git checkout main and we can switch into the main branch.

- git branch to check whether we are in the main branch.

- open -e README.md and we can see that the change made in another branch is not reflected when opening a file in the main branch.

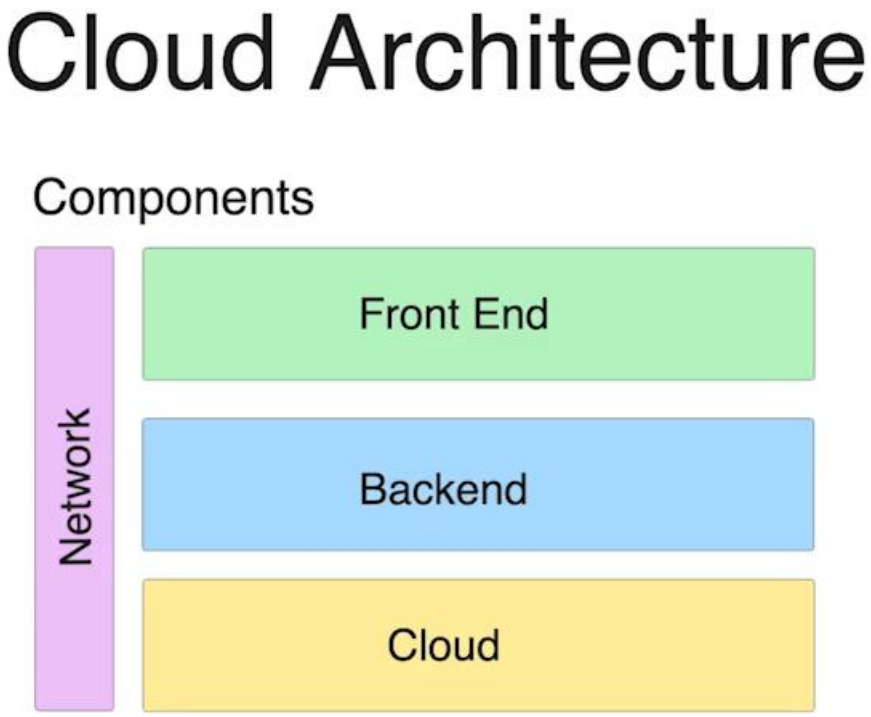
- git pull and finally we can update this change to the file in the main branch.

- open -e README.md to open the file again and finally merge the change to the file in the main branch.



제6강. Intro to Cloud Architecture

- Cloud Architecture indicates who components such as Front end, Back end, Cloud and Network are working together to build a solution and a cloud platform for creating cloud systems and environments.



1. What are the components of Cloud Architecture?

- There are four main cloud architectures which are front end, backend, cloud, network.

(1) The front end refers to the user's side of the applications and web browsers. For example, when entering Netflix.com into your browser and see the web page of the Netflix, this is the front end running in the cloud.

(2) The back end refers to the servers, data storage system and business application logic that power the functionality of a website and application. For example, in Netflix's backend, this includes the movies available in Netflix, user accounts management and also algorithms that recommend contents. The backend handles and manages data management, storage, the entire processing that supports the front-end interface to deliver the correct contents and functionality to users.

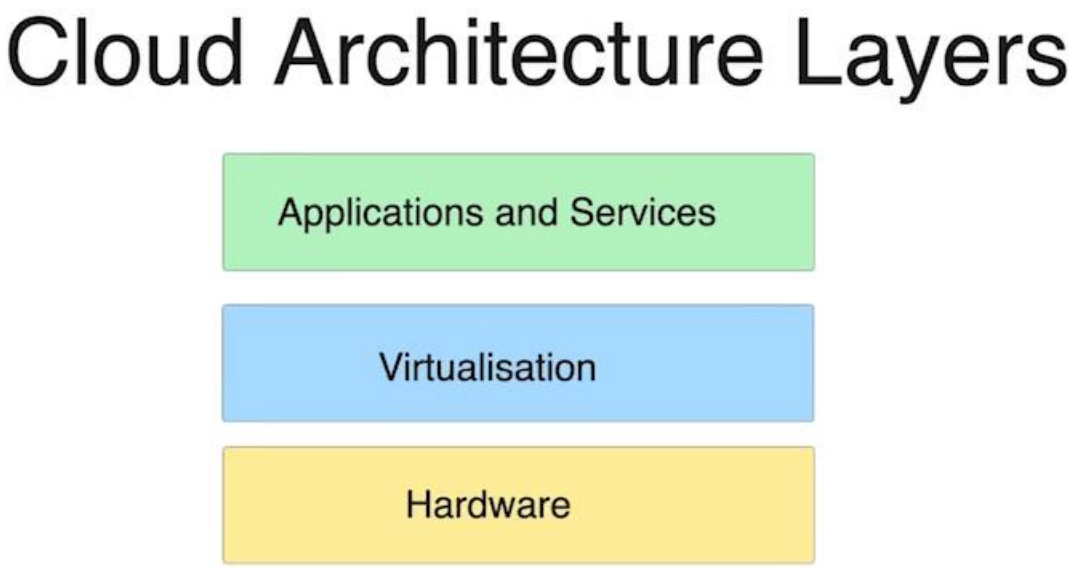
(3) Cloud-based delivery

- It includes 3 types of cloud computing models which are IaaS, PaaS and SaaS.

(4) Networking

- It refers to Internet connectivity, ensuring communication between the front end and the back end.

2. What are the Cloud Architecture Layers



- The basic cloud architecture layers are hardware, virtualization and applications and services. The hardware is the physical hardware such as servers, storage, network devices that actually power the cloud. The virtualization acts as an abstraction layer, creating virtual representations of physical computing and storage resources within the hardware layer. This enables users and applications to utilize the same resources. Virtualization allows for resource pooling that allows businesses to use and share these resources across multiple environments.

- Applications and services run on the cloud to support users' requests from the front-end user interface, offering various services based on different cloud deployment models. While applications and services are basically accessed through the front-end layer, they are connected to the virtualization layer, which virtualizes the underlying hardware resources in the cloud.

3. What is the purpose of the cloud architecture

- Cloud architecture acts as a fundamental blueprint on how cloud components will work together to form cloud environments and solutions and the core purpose of cloud architecture is to follow the best architecture principles and implement the best practices.

- The important cloud architecture design principles are composed of scalability, security, reliability, performance and cost.

(1) Scalability

- The cloud solutions designed around key cloud architecture principles design can seamlessly handle the growth in users, data and traffic.

(2) Security

- Cloud architects must design cloud architecture with security best practices and compliance requirements, ensuring data protection, privacy and adherence to industry regulations.

(3) Reliability

- Cloud architecture design ensures the resilience of systems that can withstand failures and mitigate potential downtimes by incorporating redundancies, failover strategies and disaster recovery plans, ensuring that cloud applications, services and platforms are always running with data that is secure and maintain its integrity.

(4) Performance

- Key principles of cloud architecture design focus on optimizing applications' performance in the cloud, including choosing the right database, implementing caching mechanisms and using contents delivery network (CDN) to reduce and minimize latencies and enhance user experiences. Performance is related to the content delivery speed of applications.

(5) Cost

- It is necessary to identify a cost-effective cloud architecture without compromising application performance and scalability.

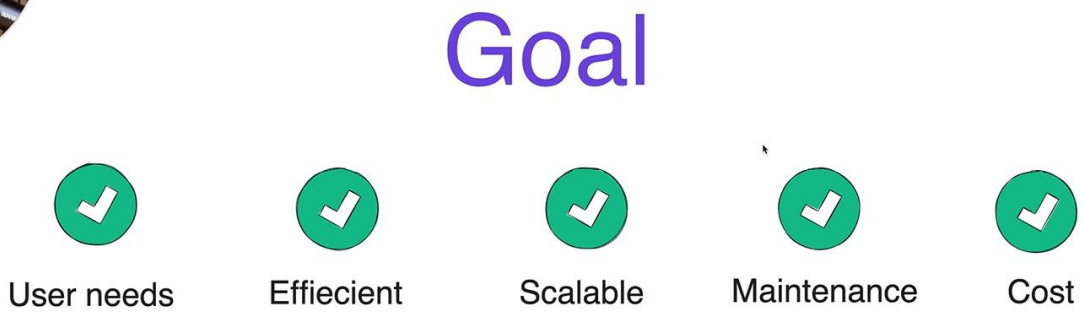
제7강. System design principle

- System design principles are related to build a software system that can scale, perform reliably and operate efficiently.

- It is essentially critical in the cloud environment to make cloud resources delivering business solutions for both business requirements and user expectations.

- System design is building and architecting systems, considering infrastructure, hardware, software and all way down to data regarding how to store and secure data. The final systems should meet users' needs, operate efficiently, easy to scale and maintain as well as cost-effective, which is the final goal of system design principles.

- From business perspectives, they usually focus on users' needs and functionality of software applications regarding how applications look like and how to design the entire users' journey by considering all products list, how should design look like and the entire payment journey whereas from engineering and architecture perspectives, they need to think about the underlying infrastructure including back-end and front-end, more related to non-functionality underneath the actual function of software applications shown to users.



- The process of system design consists of requirements analysis, architecture, component, data, interface and security

(1) Requirement includes considering both functional and non-functional requirements. Functional requirements need how this system should do to satisfy the entire customers' journey and non-functional requirements consider how the underlying resources can make the system we have designed to meet users' functional requirements regarding the final software applications.

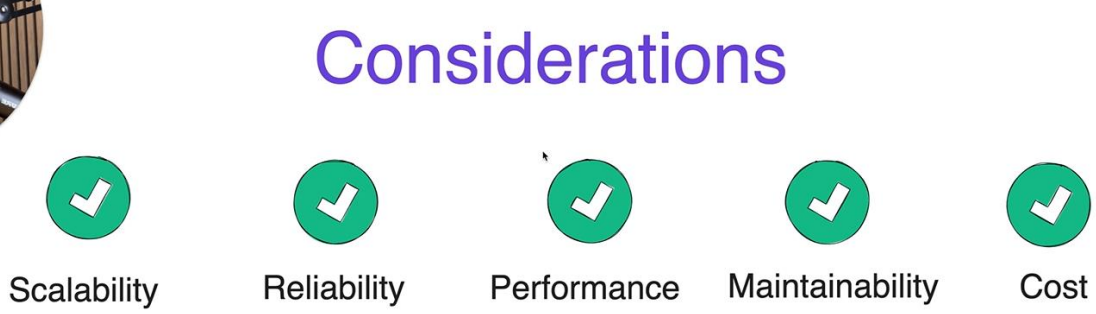
(2) Architecture stage includes designing the architecture patterns that best suit project needs, including microservices, monolithic and serverless architecture and how to different parts of systems can communicate with each other and how data can flow all way down to this architecture patterns.

(3) Component is about what we are going to utilize for meeting this architecture design that includes database, application services, users interface including technology we are going to use, framework and protocols.

(4) Data includes designing database schemas and data modules based on the requirements that includes choosing between relational and non-relational database which needs for the design of data integrity and scalability.

(5) Interface includes how different components in a new system can interact with each other and designing APIs to ensure compatibility between the new system and the rest of other systems for the seamless integration with other systems.

(6) Security should be incorporated with every measure of system including data encryption, users authentication and authorization.



(1) Scalability focuses on the accommodation of the growth in users' traffic that does not impact the overall performance of software applications that can scale based on their needs.

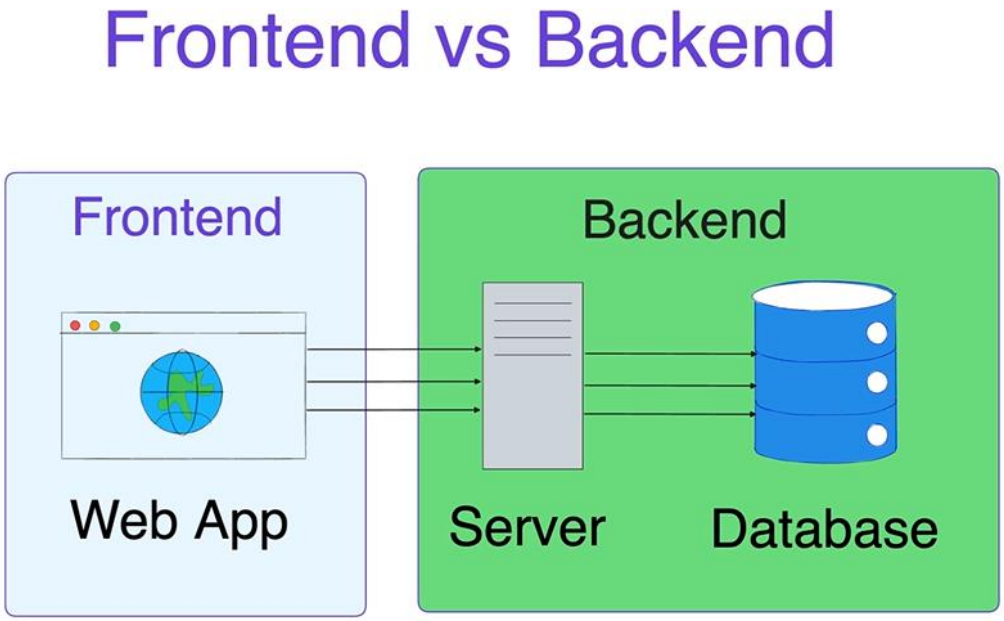
(2) Reliability includes the rapid recovery from failures, enabling software applications to continue on their operations without any major downtime.

(3) Performance is related to the speed of the applications, minimizing latency.

(4) Maintenance allows for updates and fixing bugs and any other future issue, being aware of the fact that systems we now created call any issue later on and the systems we design should be easy to fix and maintain the entire process.

(5) Cost implications should be considered for our architecture decisions by making optimal balance between the performance and cost to avoid any critical cost overrun.

제8강. Front End vs Back End

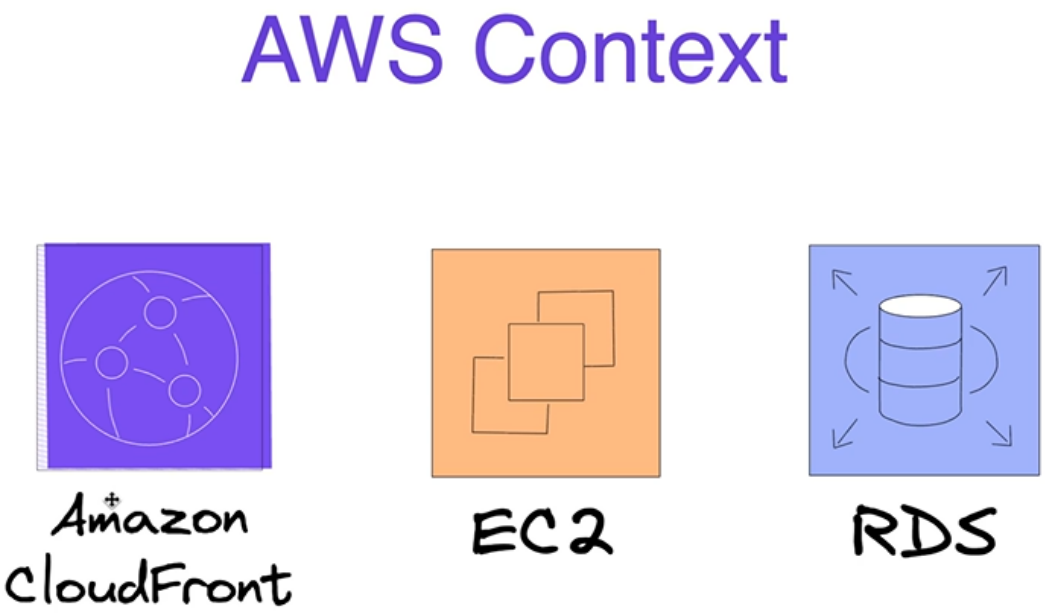


- Front end is represented by a web browser displaying a web application. The front end is the part of the web application that users interact with directly, including entering a text, clicking buttons and user interface through which users navigate the entire journey.

- The front end is responsible for sending a request to back end usually formed of HTTP request and displaying the data received from the back end to users seamlessly.

- The back end consists of server and database where the core logic of technology for web applications happens. Server receive incoming requests from the front end acting as a computer and execute the necessary operations to send the appropriate response back to the front end.

- Database store data including users' profile, contents and transactions record and server query and update the database to execute users' requests from the front end and send the response back.



- The web application, server and database can be the Amazon Cloudfront, EC2 and RDS hosted by AWS.

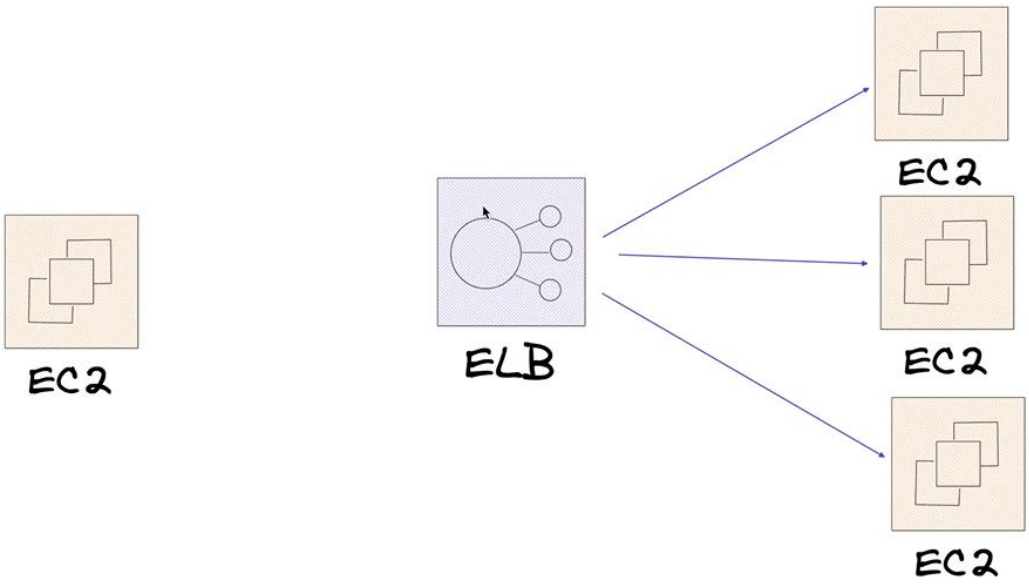
제9강. Vertical cs Horizontal Scaling

1. What is vertical and horizontal scaling

- Vertical scaling increases a computing power of single machine such as CPU, RAM, Storage. If user traffic increases, the only way to adjust to the increase in vertical scaling is to upgrade a single machine' hardware, including adding memory and storage capacity.

- Horizontal scaling increases the number of machines in a system to allow more servers with the same size to handle larger load by sharing the workload with one another while distributing traffic between them. The most crucial benefit is that we can limitlessly increase the number of servers as needed. It is related to high-availability and fault-tolerance because if a server fails, another sever can take the load instead.

- In AWS, we can do both vertical and horizontal scaling. In case of the vertical scaling, we can change the type of EC2 whereas we can use Elastic Load Balancer (ELB) and Auto-scaling group to adjust the number of EC2 instances automatically while distributing load to each machine to meet the demands.



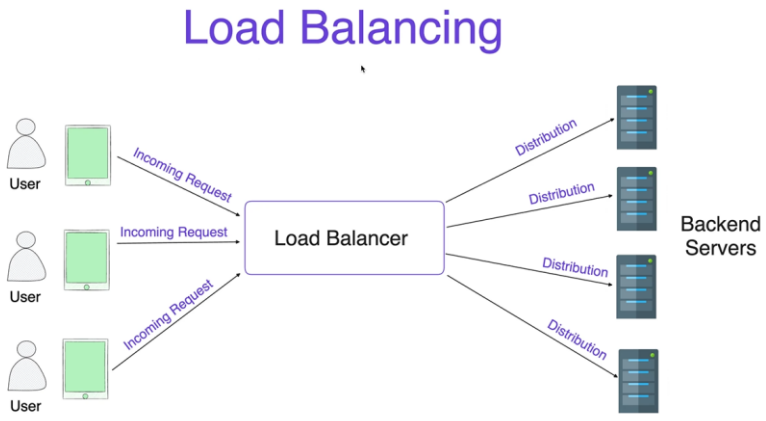
제14강 Load Balancing

- A Load balancing is a technique to distribute incoming traffic across multiple back-end servers to help applications keep running smoothly while reducing an individual server load to prevent the server from bearing too much load that can lead to the failure.

(1) How does the load balancing work?

- A load balancer receives incoming request from users and distribute them to multiple servers by using algorithm to evenly distribute traffic and optimize server resources.

- The load balancer controls the number of backend resources by scaling up and down as needed. For example, the load balancer increase the number of EC2 machines when incoming traffic increase.



(2) What is the importance of the load balancing?

1. Reduce downtime

- If incoming traffic is distributed by the load balancer across multiple servers, the risk of a single machine failure decreases. If any single machine fails, the load balancer automatically redirects traffic to another machine.

2. Improve performance

- The load balancer helps any single machine from being bottle-neck thereby improving the responsiveness and availability of software applications.

3. Scalability

- As applications are user-based growth, the load balancer can easily scale up and down backend resources. It can increase resources without interrupting the performance of applications.

4. Session persistence ('sticky session')

- During a user's interaction with frontend applications, the user's requests related to session data including user identification and shopping cart contents are directed to a specific server. This ensures that all user-related activities remain consistent by handling the session data on the same server. If a company only stores session data on a specific server, sticky session is needed to provide consistent and reliable services.

5. Health check

- The load balancer proceeds the health checks of backend servers. If servers are not responding and fail, the load balancer stop sending the traffic to them and redirect the request to other healthy servers.

6. Offload SSL decryption

- The load balancer can offload SSL decryption process from the backend servers. It means that the load balancer handles the decryption and re-encryption of data for secure HTTPS connections between clients and servers that reduces the workload of the backend servers, improving the overall efficiency of application or systems.

제15강 High Availability

- High availability is related to operational performance, ensuring that systems are always active and interactive. Users send a request to a service over the Internet and expect that they can access this service at any time without interruptions.

- A load balancer is a crucial component to achieve the high availability, receiving requests from the users through the Internet and balance the workloads by distributing them across multiple servers.

- In addition, it prevents from any server from taking a huge burden, leading to the failure and if a system is failed, it redirects the request to another server to maintain the high availability of services.

- High availability cluster is a group of servers, working together to provide continuous services. This cluster redirects incoming request to another server, if any server fails.

(2) How does the high availability work?

1. Redundancies

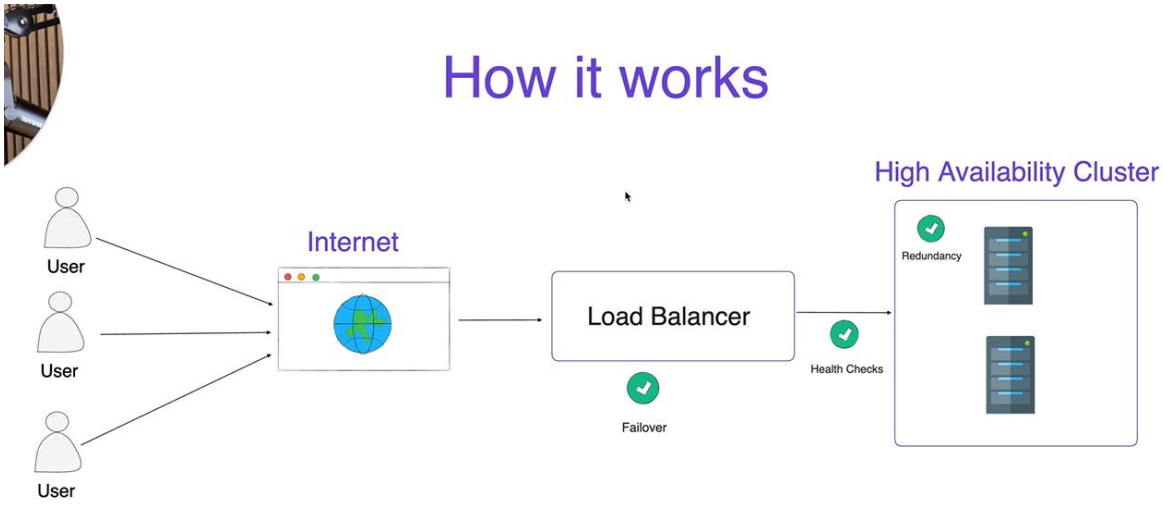
- The high availability cluster enable another server to pick up the load, if any server is down, in order to maintain the overall service performance.

2. Failover process

- The high availability system has its own failover mechanism. When the load balancer detects any server not available to respond user's request, it automatically redirect this request to other healthy servers.

3. Health checks

- Regular health checks perform health checks to ensure that all servers function correctly. If a server fails to pass the health checks, it is temporarily removed for the pool until it can be backed up and running.



(3) What is the importance of high availability

1. Reliability

- Users and businesses always expect services that can be running around 24/7. High availability systems are less likely to experience outages, improving reliability.

2. Business continuity

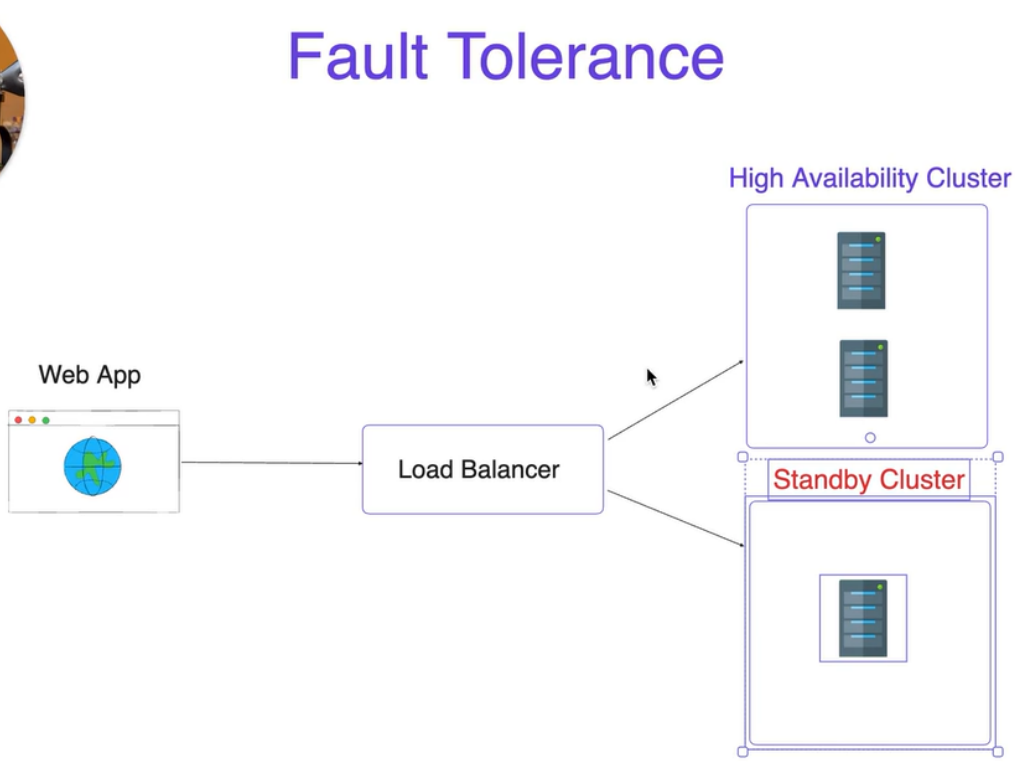
- In businesses, any downtime can lead to significant financial loss and damage on their brand and reputation. High availability systems ensure that business systems can continue on running without any interruptions.

3. Trust and Reputation

- Consistently available systems, services websites or applications build trust with customers and enhance companies' reputation.

제16강 Fault tolerance

- Fault tolerance continues systems operating in the event that some components fails, crucial for services availability and ensures that this failures do not lead to the loss of data and ruin users' experiences.



- A load balancer is a crucial component for fault tolerance by distributing incoming traffic to high availability cluster and redirect requests to remaining healthy servers, if any server is down.

- In addition, if the entire servers fail in the high availability cluster, the standby server can take the load without interrupting web application services. This provides additional layers for fault-tolerance, a kind of back-up service to maintain services availability.

(2) How does the fault tolerance work?

1. Redundancies

- System has redundant components such as servers, network paths and databases. If one system fails, another one can take over the load and they are completely different architecture and the copy of it.

2. Failover

- If the current system fails, the request is automatically redirected to redundant servers.

3. Health checks

- Health checks enable any unhealthy system or server to be protected and respond to this failure as soon as possible.

(3) What is the difference between fault-tolerance and high-availability?

- High availability focuses on minimizing downtime by responding to unexpected failures. High available systems are designed to be recovered from any failure to ensure services to be available as much as possible. It is all about uptime and high available systems are redundant and active concurrently.

- Fault-tolerance systems are designed to prevent the downtime entirely and seamlessly recovered from any server failure by other remaining servers taking over the load without manual interventions.

- Fault tolerance systems also have redundant components that are not just available but also operational, meaning that they are not just present as backups but fully functional and ready to take over instantly without any delay.

- Fault tolerance systems can also involve geographic distribution where redundant components are relocated in different data centers or regions to protect against localization disaster.

- Despite outages, high availability systems can be restored to functionality through rapid recovery mechanisms. However, there might be a brief downtime during recovery. Fault tolerance is about the system's ability to continue operating without interruption, even if a specific server fails. High availability is less complicated and more cost-effective than fault-tolerance, which is why most companies prefer high availability systems for the balance of reliability and cost.

WEEK 2 PROJECTS

Week2. Git & Cloud Architecture

Exercise 1: AWS CLI Setup and Configuration

Objective: Learn how to configure the AWS CLI.

1. Install the AWS CLI following the official AWS documentation.

2. Configure the AWS CLI with an IAM user credential using aws configure. Enter your AWS Access Key ID, Secret Access Key, default region name, and default output format.

---

1. Log into an IAM user page for 'JihyunAWS' and create an Access Key and select the Command Line Interface option for the Access Key.

2. Open Command Prompt or PowerShell.

3. Run the configuration command.

```
C:\Users\sjh50>aws configure
AWS Access Key ID [****************SSNF]: AKIA4AQ3TX2SHTEGPLLJ
AWS Secret Access Key [****************/7xZ]: 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
Default region name [ap-northeast-2]: ap-northeast-2
Default output format [None]: json
```

Exercise 2. Listing S3 Buckets

Objective: Practice listing Amazon S3 buckets using the CLI.

1. Use the aws s3 ls command to list all the S3 buckets under you AWS account.

2. Identify how many buckets are currently created and document their names.

---

1. List all the S3 buckets

```
C:\Users\sjh50>aws s3 ls

C:\Users\sjh50>
```

If the 'aws s3 ls' command didn't return anything, it is likely that there are no S3 buckets currently.

Exercise 3. Creating an S3 Bucket

Objective: Learn to create an S3 bucket using the CLI.

1. Create a new S3 bucket using aws s3 mb s3://<your-bucket-name>, replacing <your-bucket-name> with a unique name.

2. Verify the bucket was created successfully by listing all buckets.

```
C:\Users\sjh50>aws s3 mb s3://JihyunS3bucket
make_bucket failed: s3://JihyunS3bucket An error occurred (InvalidBucketName) when calling the CreateBucket operation: T
he specified bucket is not valid.

C:\Users\sjh50>aws s3 mb s3://jihyuns3bucket
make_bucket: jihyuns3bucket

C:\Users\sjh50>aws s3 ls
2024-08-25 17:18:38 jihyuns3bucket

C:\Users\sjh50>
```

1. When I created the bucket name as JihyunS3bucket, the error occurred because the bucket name I created didn't meet Amazon S3's naming requirements.

2. S3 bucket Naming Rules:

1) Length: Bucket names must be between 3 and 63 characters long.

2) Bucket names only allow lowercase letters, numbers, hyphens ('-') and periods (' . ').

3) Bucket names must start and end with a lowercase letter or a number.

4) No Uppercase or Underscores ('_')

5) No IP-Like Format.

Exercise 4. Uploading a File to S3

Objective: Practice uploading a file to an S3 bucket.

1. Create a text file named test.txt and add some content to it.

2. Upload the file to your previously created S3 bucket using aws s3 cp test.txt s3://<your-bucket-name>/.

3. Verify the file was uploaded successfully by listing the contents of your bucket.

---

1. Create a file directly from the command line:

```
C:\Users\sjh50>echo "This is a test file." > test.txt

C:\Users\sjh50>type test.txt
"This is a test file."

C:\Users\sjh50>notepad test.txt

C:\Users\sjh50>
```

2. Upload the file to your S3 bucket and verify the file was successfully uploaded.

```
C:\Users\sjh50>aws s3 cp test.txt s3://jihyuns3bucket
upload: .\test.txt to s3://jihyuns3bucket/test.txt

C:\Users\sjh50>aws s3 ls s3://jihyuns3bucket
2024-08-25 17:26:34         25 test.txt

C:\Users\sjh50>
```

Commands:

1) ls: list files or directories.

2) cp: copy the file from one location to another.

3) echo: create a new file and add a content to it.

4) File size (e.g., 25): refers to the size of the file in bytes.

Project – AWS CLI – IAM User

Week 2 – Git & Cloud Architecture

Exercise: Creating an IAM User with the AWS CLI

Objective: Learn to create a new IAM user and attach a policy granting them specific permissions using the AWS CLI.

Part 1: Create a New IAM User

1. Create the IAM user: Use the aws iam create-user command to create a new IAM user. Replace <UserName> with the desired name for the new user.

aws iam create-user --user-name <UserName>

```
C:\Users\sjh50>aws iam create-user --user-name JennyWithCloud
{
    "User": {
        "Path": "/",
        "UserName": "JennyWithCloud",
        "UserId": "AIDA4AQ3TX2SEDNZU333C",
        "Arn": "arn:aws:iam::825765379748:user/JennyWithCloud",
        "CreateDate": "2024-08-26T01:45:44+00:00"
    }
}
```

2. Verify the user creation: List all IAM users to ensure your new user was created successfully.

aws iam list-users

```
C:\Users\sjh50>aws iam list-users
{
    "Users": [
        {
            "Path": "/",
            "UserName": "JennyWithCloud",
            "UserId": "AIDA4AQ3TX2SEDNZU333C",
            "Arn": "arn:aws:iam::825765379748:user/JennyWithCloud",
            "CreateDate": "2024-08-26T01:45:44+00:00"
        },
        {
            "Path": "/",
            "UserName": "JihyunAWS",
            "UserId": "AIDA4AQ3TX2SAGOD62B4U",
            "Arn": "arn:aws:iam::825765379748:user/JihyunAWS",
            "CreateDate": "2024-08-25T23:40:52+00:00",
            "PasswordLastUsed": "2024-08-25T23:44:40+00:00"
        }
    ]
}
```

Part2: Attach a policy to the New User

To grant permissions to the user, you will attach a policy. For this exercise, you will attach the AmazonS3ReadOnlyAccess policy, allowing the user to list and read objects in S3 buckets.

1. Attach the policy: Use the aws iam attach-user-policy command to attach the AmazonS3ReadOnlyAccess policy to your newly created user. Replace <UserName> with the name of the user you created.

aws iam attach-user-policy - -user-name <UserName> --policy-arn

arn:aws:iam: :aws:policy/AmazonS3ReadOnlyAccess

```
C:\Users\sjh50>aws iam attach-user-policy --user-name JennyWithCloud --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOn
lyAccess
```

2. Verify the policy attachment: Check the attached polices of the user to ensure the AmazonS3ReadOnlyAccess policy is attached.

aws iam list-attached-user-policies - -user -name <UserName>

```
C:\Users\sjh50>aws iam list-attached-user-policies --user-name JennyWithCloud
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonS3ReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
        }
    ]
}
```

Part3: (Optional) Enable Programmatic Access

If you want your IAM user to interact with AWS services through AWS CLI or SDKs, you will need

to create access keys for programmatic access.

1. Create access keys: Generate new access keys for the IAM user.

aws iam create-access-key - -user -name <UserName>

2. Store the access keys: Make sure to securely store the AccessKeyID and SecretAccessKey returned by the command. These credentials are needed for the user to authenticate with AWS programmatically.

```
C:\Users\sjh50>aws iam create-access-key --user-name JennyWithCloud
{
    "AccessKey": {
        "UserName": "JennyWithCloud",
        "AccessKeyId": "AKIA4AQ3TX2SEADW24ZG",
        "Status": "Active",
        "SecretAccessKey": "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",
        "CreateDate": "2024-08-26T01:56:39+00:00"
    }
}
```

- An ARN (Amazon Resource Name) is a unique identifier used by AWS to reference various resources within your AWS environment. ARN are used to specify and identify resources across all of AWS's services, such as IAM roles, S3 buckets, DynamoDB tables, Lambda functions and more.

Project – Design a Basic Architecture

Week 2 – Git & Cloud Architecture

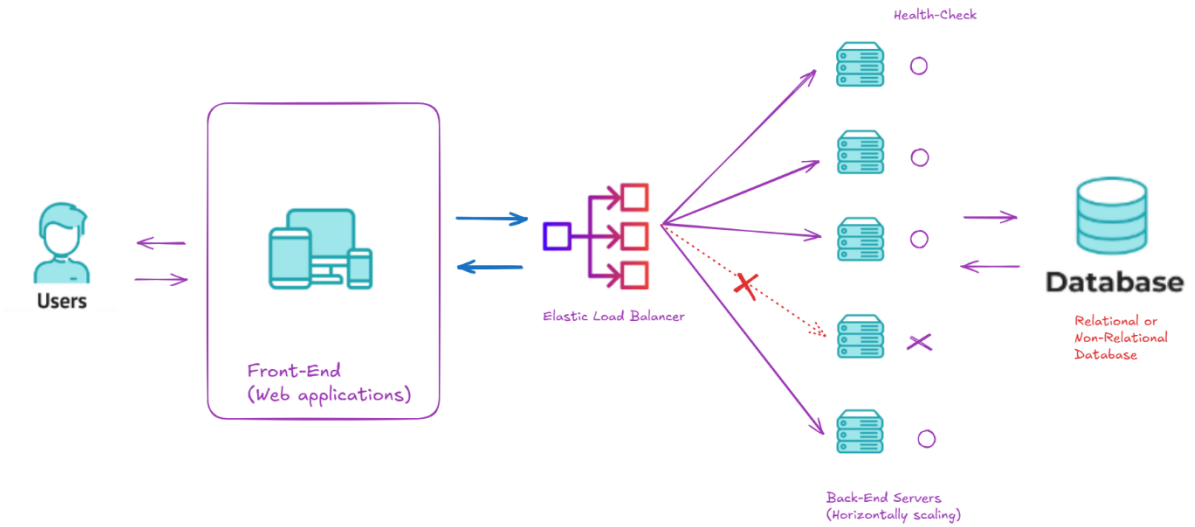
Objective: Develop a conceptual design for a cloud infrastructure supporting a web application.

Task 1: Identify Components

1) Front-End Web browser or Web application

2) Load Balancer

3) Back-End Server



- Users interact with a web application or web browser to send a request and receive feedback. In

the AWS, the Elastic Load Balancer (ELB) receives requests through the Front-end and evenly distribute them across Back-end servers to prevent a specific server from receiving huge loads, leading to the failure to achieve high availability of the web application. ELB also checks the health of each server and if there is any unhealthy server, ELB automatically redirects the load to another server. Back-end servers interact with RDS to extract necessary information needed to deliver users' requests and send the responses back to the front-end, which displays the results to users. Back-end servers can automatically scale horizontally by using auto-scaling group (ASG) when users' traffic increases, that is also related to the high-available and reliable web service for users.