Tecnológico de Monterrey Campus Santa Fe

System Documentation

Gabriel Rodríguez De Los Reyes A01027384

Mariel Gómez Gutiérrez A01275607

Pablo Banzo Prida A01782031

Santiago Rodriguez Palomo A01025232

Grupo 501

Final Assessment

June 6th, 2025

Project description.....2
Architecture.....2
Description of Services.....3
Network.....4
Configurations.....5

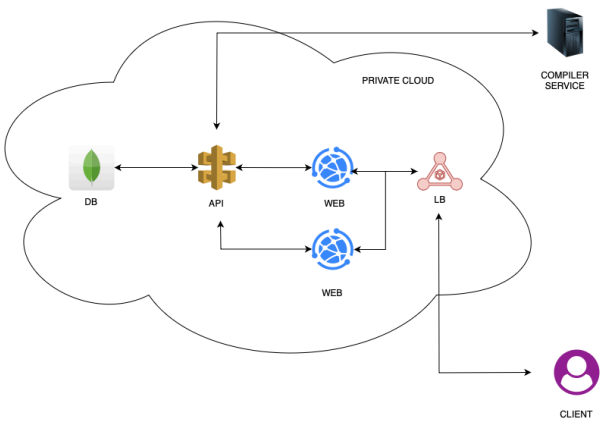
Project description

At Compilo, we've built the ultimate platform for learning how to code because we believe programming education should be accessible, engaging, and effective for everyone. Whether you're taking your first steps into the world of coding or you're a developer looking to sharpen your existing skills, we provide all the tools and resources you need to succeed in one comprehensive platform. Our approach centers around three core pillars that set us apart: our interactive browser-based editor delivers syntax highlighting and real-time feedback so you can code seamlessly without any setup hassles, our instant testing feature lets you run your code against test cases and see results immediately to accelerate your learning, and our beginner-friendly design ensures new coders get the helpful hints and clear explanations they need to build confidence. We've designed Compilo to eliminate the barriers that often prevent people from learning to code, creating an environment where anyone can start their programming journey and truly conquer the world of software development.

As part of this adventurous implementation, our diverse knowledge was used to host the platform on a private cloud. We will run part of our platform on a server, and host other architecture components. To achieve this task successfully, it is necessary to properly configure every element and keep in mind its functionality, so that all of them can be integrated seamlessly and deliver the expected result. In this document, we will dig deeper into the implementation of these components, that architecture design, test and results.

Architecture

Based on the given requirements, specifications and limitations, we designed an architecture that can fulfill all of the restrictions and expectations. We expect to have the best performance by having a unified system that works seamlessly achieving all of our objectives. We are integrating different services that are not only compatible with each other, but also have an easy integration that can allow them to work with many others in case that we need to scale up our services.



Description of Services

MongoDB: MongoDB serves as our primary database solution, offering a flexible NoSQL, open-source, document-oriented architecture. This database system scales efficiently from simple, small-scale implementations to complex, enterprise-level deployments. MongoDB provides advanced capabilities including interactive search, vector search, and real-time stream processing. Its inherent flexibility enables seamless integration with diverse services and applications, while its intuitive graphical user interface simplifies database management. The extensive MongoDB community contributes comprehensive documentation, tutorials, and implementation guides, facilitating straightforward deployment and configuration.

Docker: Docker provides an open platform for application development, deployment, and execution through containerization technology. This platform enables applications to be packaged with their dependencies into portable containers, ensuring consistent performance across diverse environments. Docker containers operate seamlessly on developer workstations, physical servers, virtual machines, cloud platforms, or hybrid infrastructure configurations. The platform integrates natively with CI/CD workflows, standardizing development environments and enabling efficient application distribution and testing. Container lifecycle management is simplified through Docker's comprehensive API and command-line interface.

Go: Go is an open-source programming language developed and maintained by Google, featuring a robust standard library and extensive ecosystem of development tools. The language excels in multiple implementation scenarios, including command-line interface development, web application creation, DevOps automation, and API construction—the latter being critical for our project requirements. Go delivers rapid compilation times that support iterative development workflows while maintaining low memory overhead. The language integrates seamlessly with containerization platforms like Kubernetes and Docker. Go's standard library provides native database connectivity with support for most SQL databases, and MongoDB offers official Go driver support with advanced features including client-side field encryption and comprehensive error handling through the ServerError interface.

Next.js: Next.js is a comprehensive full-stack framework built on React that enables the development of high-performance web applications. The framework includes built-in optimizations for enhanced user experience and Core Web Vitals compliance, featuring dynamic HTML streaming and support for the latest React capabilities. Next.js automatically configures underlying development tools and provides essential features such as advanced data fetching mechanisms and server-side code execution. The framework supports Incremental Static Regeneration (ISR) on a per-page basis when static content optimization is required.

NGINX: NGINX functions as a high-performance HTTP web server, reverse proxy, content cache, load balancer, TCP/UDP proxy server, and mail proxy server. In our architecture,

Nginx serves primarily as a load balancer to distribute incoming traffic efficiently across multiple backend servers.

Flask: Flask is a lightweight Web Server Gateway Interface (WSGI) framework that has become the standard for Python web application development. The framework maintains a minimalist core design while remaining highly scalable and supporting extensive functionality through its plugin ecosystem. Flask's emphasis on code readability and developer-friendly design makes it an optimal choice for implementing our Python-based compiler service, providing the necessary web interface for our compilation scripts.

OpenStack Cloud: OpenStack represents the most widely adopted open-source cloud computing platform globally, managing extensive pools of compute, storage, and networking resources. The platform comprises multiple integrated components including Nova (compute), Zun (container management), Swift (object storage), and Octavia (load balancing), among others. OpenStack supports diverse programming languages including Go and Python, and offers dual management interfaces: a comprehensive web-based dashboard for administrative control and RESTful APIs for programmatic resource provisioning and management.

Network

Our network is composed of several important elements. The first element is a shared switch, which is being used by all of the teams. We are using port number 23 (access port) and port number 24 (trunk ports). To this switch, we connected our server, and our cloud. This switch is connected to our shared router, now only shared with three teams. The router will allow us to connect to the internet, so we can run our demo in any campus building that has the “Tec” Network.

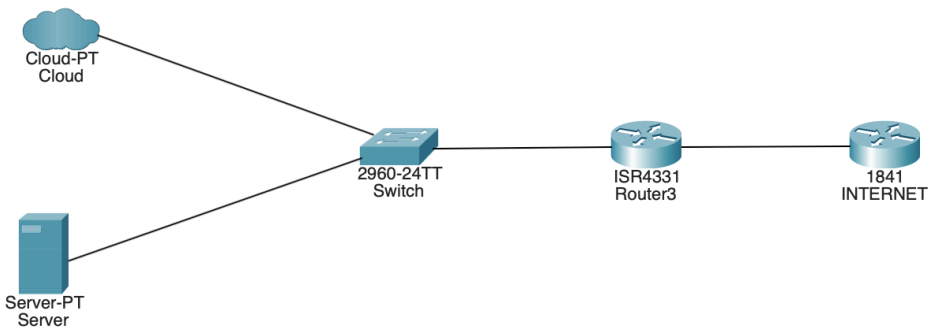


Image. Network topology

Configurations

The router given to us was Router number 3, to which we assigned the “R-INF-3” host name. For security purposes, we enabled secret 9, which will hash encrypt our password, using script hashing. DHCP (Dynamic Host Configuration Protocol) was included so we can automatically get an IP assigned; our pool name was “VLAN30” just for consistency purposes. The network range will be from 172.16.30.0 to 172.16.30.254 taking into consideration its 24 mask. When implementing this protocol, it is necessary to designate a default router, which for the project will be 172.16.30.1. After basic configuration was made, we proceeded to more detailed features. Our trunking protocol was set to 1, which is the most basic and classic one.

We implemented NAT translation after encapsulating our VLANS. On the GigabitEthernet Interface 0/0/0.30, we configured NAT inside, and on GigabitEthernet Interface 0/0/1 we configured NAT outside, because that's the side that will have the internet connection. Our NAT Ips were also set up to be extendable, which allows the same mapping to be used from multiple inside interfaces. Thanks to this configuration , we will be able to use only one router physical port, and translate our IP when going through the outside interface. We considered it important to configure HTTP services, so we enabled the HTTP Secure Server. Finally for our load balancer implementation, we needed to implement PAT.

Final configuration del router:

Building configuration...

Current configuration : 7459 bytes

!

! Last configuration change at 20:36:47 UTC Fri Jun 6 2025

!

version 17.6

service timestamps debug datetime msec

service timestamps log datetime msec

service password-encryption

! Call-home is enabled by Smart-Licensing.

service call-home

```
platform qfp utilization monitor load 80
platform punt-keepalive disable-kernel-core
!
hostname R-INF-3
!
boot-start-marker
boot-end-marker
!
!
enable secret 9 $9$jNpu7SO0Z181ak$yTtcH8bz0Pfjqp.ZD1cfP.tVMy7Uai1qOt.r1dlpBEU
enable password 7 0257530859
!
no aaa new-model
!
!
!
!
!
!
!
ip domain name mydomain.local
ip dhcp excluded-address 172.16.30.3
!
ip dhcp pool VLAN30
 network 172.16.30.0 255.255.255.0
 default-router 172.16.30.1
!
!
```

```
!
login on-success log
!
!
!
!
!
!
!
subscriber templating
vtp version 1
multilink bundle-name authenticated
!
!
!
crypto pki trustpoint TP-self-signed-654147102
 enrollment selfsigned
 subject-name cn=IOS-Self-Signed-Certificate-654147102
 revocation-check none
 rsakeypair TP-self-signed-654147102
!
crypto pki trustpoint SLA-TrustPoint
 enrollment pkcs12
 revocation-check crl
!
!
crypto pki certificate chain TP-self-signed-654147102
 certificate self-signed 01
```

3082032E 30820216 A0030201 02020101 300D0609 2A864886 F70D0101 05050030

30312E30 2C060355 04031325 494F532D 53656C66 2D536967 6E65642D 43657274

69666963 6174652D 36353431 34373130 32301E17 0D323530 36303232 30343334

385A170D 33353036 30323230 34333438 5A303031 2E302C06 03550403 1325494F

532D5365 6C662D53 69676E65 642D4365 72746966 69636174 652D3635 34313437

31303230 82012230 0D06092A 864886F7 0D010101 05000382 010F0030 82010A02

82010100 A55EB104 EF4D7A48 7EEE93BA 698DCA83 98F880F4 788A2D32 8D3CA8C6

A6C1A084 7CD8CD9B 033257AA 7664897E CBCD7EEA A4E4C60E A2BFA800 CC291056

30C93FB2 3475F133 E96703E4 FDB3C2FC F15FB583 8A2AC740 A457ACF5 D31A6957

902AD474 CF96D1C1 0703E37F E05E5D08 F80A00C0 12912E45 6D5424F2 2B5AF611

B1A54BCB EE845DD1 5E5F6DF6 AABD528A 0A1FA4C2 DB7050F7 5A8EFF83 F16C65CA

42F109A6 B51FBA87 3834854B 3338112D 2BA0F2CF B3169B2B E7F4BC4B AFBF366E

6BE561D9 4CEF80D8 F41F6725 BCA84F0F 7534543A 3D6AD31C FD7642E9 8E51461E

E51B88F9 9A9B8E19 3925693E 0485704D 9070AEA4 1A86B1BA 7FBFF8C8 A2B0660B

F4ACD7ED 02030100 01A35330 51300F06 03551D13 0101FF04 05300301 01FF301F

0603551D 23041830 1680146A D42E3325 877384E0 83488969 22A13CEE 13CE2630

1D060355 1D0E0416 04146AD4 2E332587 7384E083 48896922 A13CEE13 CE26300D

06092A86 4886F70D 01010505 00038201 010019DF A509770F AA12402E FF684410

E4B3E09C CE2CA6FB FFD9A551 34748AD5 C9A71987 478E9733 D0B338D0 C67767E9

DB69631C 9A14F212 B11EC107 6CFD8808 D2C13B30 70659D9C 5CE3377E 4D23F42C

935F9B9C 0EFC4B33 A199AA99 01650954 369F7FB3 A0CD41BA 36B74FCD 27273FE7

91AE6EDD 8B8B5FC5 32919269 25213FCD 06AF8199 2D7472DE 6DF7B486 ECEC9162

D36706CE FFAFD2A0 2F2777E5 BE8F2C66 45EFFCB2 B509A15F B23D9086 2EE0BB49

4144DC0B 0A496616 431D0CB2 71667D7A BB247F9F 507D5DA1 EA4B125F A8F3AEC2

A0DE3D17 7DFAE880 7D61073A 83621E92 5B1A5EDA BA12628C 0B57787E C9B72FAA

C1D1947F D95682B3 BB6E9F34 67EE5A4E 2D10

quit

crypto pki certificate chain SLA-TrustPoint

certificate ca 01

30820321 30820209 A0030201 02020101 300D0609 2A864886 F70D0101 0B050030

32310E30 0C060355 040A1305 43697363 6F312030 1E060355 04031317 43697363

6F204C69 63656E73 696E6720 526F6F74 20434130 1E170D31 33303533 30313934

3834375A 170D3338 30353330 31393438 34375A30 32310E30 0C060355 040A1305

43697363 6F312030 1E060355 04031317 43697363 6F204C69 63656E73 696E6720

526F6F74 20434130 82012230 0D06092A 864886F7 0D010101 05000382 010F0030

82010A02 82010100 A6BCBD96 131E05F7 145EA72C 2CD686E6 17222EA1 F1EFF64D

CBB4C798 212AA147 C655D8D7 9471380D 8711441E 1AAF071A 9CAE6388 8A38E520

1C394D78 462EF239 C659F715 B98C0A59 5BBB5CBD 0CFEBEA3 700A8BF7 D8F256EE

4AA4E80D DB6FD1C9 60B1FD18 FFC69C96 6FA68957 A2617DE7 104FDC5F EA2956AC

7390A3EB 2B5436AD C847A2C5 DAB553EB 69A9A535 58E9F3E3 C0BD23CF 58BD7188

68E69491 20F320E7 948E71D7 AE3BCC84 F10684C7 4BC8E00F 539BA42B 42C68BB7

C7479096 B4CB2D62 EA2F505D C7B062A4 6811D95B E8250FC4 5D5D5FB8 8F27D191

C55F0D76 61F9A4CD 3D992327 A8BB03BD 4E6D7069 7CBADF8B DF5F4368 95135E44

DFC7C6CF 04DD7FD1 02030100 01A34230 40300E06 03551D0F 0101FF04 04030201

06300F06 03551D13 0101FF04 05300301 01FF301D 0603551D 0E041604 1449DC85

4B3D31E5 1B3E6A17 606AF333 3D3B4C73 E8300D06 092A8648 86F70D01 010B0500

03820101 00507F24 D3932A66 86025D9F E838AE5C 6D4DF6B0 49631C78 240DA905

604EDCDE FF4FED2B 77FC460E CD636FDB DD44681E 3A5673AB 9093D3B1 6C9E3D8B

D98987BF E40CBD9E 1AECA0C2 2189BB5C 8FA85686 CD98B646 5575B146 8DFC66A8

467A3DF4 4D565700 6ADF0F0D CF835015 3C04FF7C 21E878AC 11BA9CD2 55A9232C

7CA7B7E6 C1AF74F6 152E99B7 B1FCF9BB E973DE7F 5BDDEB86 C71E3B49 1765308B

5FB0DA06 B92AFE7F 494E8A9E 07B85737 F3A58BE1 1A48A229 C37C1E69 39F08678

80DDCD16 D6BACECA EEBC7CF9 8428787B 35202CDC 60E4616A B623CDBD 230E3AFB

418616A9 4093E049 4D10AB75 27E86F73 932E35B5 8862FDAE 0275156F 719BB2F0

D697DF7F 28

quit

!

!

license udi pid C8200L-1N-4T sn FJC282212SQ

memory free low-watermark processor 67708

!

diagnostic bootup level minimal

!

spanning-tree extend system-id

!

username admin privilege 15 password 7 02050D480809

username santi privilege 15 secret 9 9Y1kO0fLK3s.as.$8N5up8795lIVF.MuVeNXpTIdZE6itQ6OPRBKsFobwcQ

```
username tena privilege 15 secret 9 $9$0m1V..SurofxK.$z7aqHJM1mQEeP6l65e1Luvmp96APr50XZwdHhS4PfXs
!
redundancy
 mode none
!
!
!
!
!
!
!
!
!
interface GigabitEthernet0/0/0
 no ip address
 negotiation auto
!
interface GigabitEthernet0/0/0.30
 encapsulation dot1Q 30
 ip address 172.16.30.1 255.255.255.0
 ip nat inside
!
interface GigabitEthernet0/0/1
 ip address dhcp
 ip nat outside
 negotiation auto
!
interface GigabitEthernet0/0/2
```

```
 no ip address
 shutdown
 negotiation auto
!
interface GigabitEthernet0/0/3
 no ip address
 shutdown
 negotiation auto
!
interface Serial0/1/0
 no ip address
 shutdown
!
interface Serial0/1/1
 no ip address
 shutdown
!
ip http server
ip http authentication local
ip http secure-server
ip forward-protocol nd
ip nat inside source static tcp 172.16.30.26 8000 10.49.12.48 304 extendable
ip nat inside source static tcp 172.16.30.147 27017 10.49.12.48 305 extendable
ip nat inside source static tcp 172.16.30.156 3000 10.49.12.48 333 extendable
ip nat inside source static tcp 172.16.30.189 80 10.49.12.48 2025 extendable
ip nat inside source static tcp 172.16.30.3 22 10.49.12.48 3000 extendable
ip nat inside source static tcp 172.16.30.88 3001 10.49.12.48 3001 extendable
ip nat inside source static tcp 172.16.30.3 3002 10.49.12.48 3002 extendable
```

```
ip nat inside source static tcp 172.16.30.149 80 10.49.12.48 3003 extendable
ip nat inside source list 30 interface GigabitEthernet0/0/1 overload
ip route 192.168.200.0 255.255.255.0 172.16.30.254
ip ssh version 2
!
!
!
ip access-list standard 30
 10 permit 172.16.30.0 0.0.0.255
!
!
!
control-plane
!
!
line con 0
 stopbits 1
line aux 0
line vty 0 4
 password 7 0055445556
 login local
 transport input ssh
line vty 5 14
 password 7 0055445556
 login
 transport input ssh
!
call-home
```

! If contact email address in call-home is configured as sch-smart-licensing@cisco.com

! the email address configured in Cisco Smart License Portal will be used as contact email address to send SCH notifications.

contact-email-addr sch-smart-licensing@cisco.com

profile "CiscoTAC-1"

active

destination transport-method http

!

!

!

!

!

!

end

References

mongo package - go.mongodb.org/mongo-driver/mongo - Go Packages. (2025). Go.dev. https://pkg.go.dev/go.mongodb.org/mongo-driver/mong

The Go Programming Language. (2025). Go.dev. https://go.dev/

Vercel. (2025). *Introduction*. Nextjs.org. https://nextjs.org/docs

MongoDB: The World's Leading Modern Database. (2024). MongoDB. https://www.mongodb.com/

(2025). Openstack.org. https://www.openstack.org/software/

What is Docker? (2024). Docker Documentation. https://docs.docker.com/get-started/docker-overview/