Global Yugabyte Database Deployment into Google Cloud (Part 2)

Restartable Yugabyte Nodes on Compute Instances

Christoph Bussler

7 min read · Just now

When a compute instance restarts for any reason, the Yugabyte master and tserver running on the compute instance should automatically restart as well. This blog describes one approach to accomplish the automatic restart of Yugabyte's components.

The previous blog Deployment into Google Cloud (Part 1) followed the Yugabyte instructions that starts Yugabyte master and tserver with command line parameters on a compute instance: Three+ Data Center (TDC). However, this approach of using command line parameters does not restart Yugabyte master or tserver when restarting the compute instance.

This blog describes an approach based on the combination of compute instance startup script and Yugabyte master and tserver configuration files.

Approach: multi-phase compute instance startup script and Yugabyte configuration files

When a compute instance is created, Yugabyte needs to be installed first.

Once Yugabyte is installed, and once configuration files for the Yugabyte master and tserver containing the required startup parameter are available on the compute instance's disk then the Yugabyte master and tserver can be started: the configuration files contain the startup parameter that originally were supplied as command line parameters.

In the following the Yugabyte configuration files, the compute instance startup script and the overall process are outline in more detail.

In a nutshell, first, the startup script installs Yugabyte and required prerequisites, second, when the configuration files are available on the compute instance, the startup script starts Yugabyte master and tserver on the compute instance.

For the 37 region deployment, the phases of the startup script are executed on each compute instance, aka, 37 times.

Yugabyte configuration files

Instead of specifying command line parameters when starting Yugabyte master or tserver, the parameters can be stored in configuration files as outline here: master and tserver — one configuration file for Yugabyte master, and one configuration file for Yugabyte tserver.

The following configuration file is an example for a Yugabyte master (taken from master):

```
--
master_addresses=172.151.17.130:7100,172.151.17.220:7100,172.151.17.140:7100
--rpc_bind_addresses=172.151.17.130:7100
--fs_data_dirs=/home/centos/disk1,/home/centos/disk2
--placement_cloud=aws
--placement_region=us-west
--placement_zone=us-west-2a
```

When starting Yugabyte master or tserver the start command refers to the corresponding configuration file.

The following example shows starting a Yugabyte master as a process with the parameters in the configuration file `master.conf` and the output written to `yb-master.out`:

```
./bin/yb-master --flagfile master.conf >& yb-master.out &
```

The configuration files are stored on the compute instance's disk. This ensures that after a restart of the compute instance the configuration files are available and the startup script starts Yugabyte master and tserver.

The benefit of this approach is that the Yugabyte master and tserver configuration is stored as files on the compute instance's disk and stopping of a compute instance does not loose the Yugabyte parameters specific to the compute instance. Starting a compute instance — that automatically executes the startup script — enables check for the configuration files on the compute instance's disk and subsequently an automatic start of Yugabyte master and tserver.

Compute instance startup script

A compute instance startup script executes when a compute instance is created and every time a compute instance is restarted. In addition, it is possible to trigger the rerun of a startup script without having to restart the compute instance.

In context of running Yugabyte on compute instances, a startup script has to provide two distinct features

1. **Installation of Yugabyte and any required software or packages**. This ensures that the startup script downloads and/or updates every needed dependency, including the Yugabyte installation itself.
2. **Starting of a Yugabyte master and tserver on a compute instance**. When the Yugabyte software is installed, and when the configuration files for the Yugabyte master and tserver are on the compute instance's disk, the startup script launches Yugabyte master and tserver.

Each functionality in the startup script is conditionally executed:

- If Yugabyte is not installed, then first install or update all dependencies, and second download the Yugabyte installation and install Yugabyte.
- If Yugabyte is installed, and the configuration files for Yugabyte master and tserver are on the disk, then launch Yugabyte master and tserver.

This conditional execution ensures that no matter when the startup script is executed, and no matter how often it is executed, the correct state is accomplished.

To ensure that each compute instance executes the exact some startup script, the approach is to store the startup script in the Google Cloud's projects meta data.

Configuration file creation and upload

One question remains: how are the Yugabyte configuration files written to the compute instance's disk?

As you can see from the example above, a configuration file requires to enumerate the (internal) IP addresses of all the compute instances that run a Yugabyte master. This means that before the configuration files can be written, all the compute instance IP addresses must be known.

With the following command all compute instances can be retrieved:

```
gcloud compute instances list
```

Once the compute instances' information is available, their internal IP addresses can be extracted. Once those are known, the configuration file can be created and written to all compute instances' disks.

This is done by a script that I am currently running on my laptop after the compute instances are created. Of course, this must be automated for a production environment, and clearly not on a laptop :-).

Once the two configuration files, one for Yugabyte master, and one for Yugabyte tserver are written, the startup script on the compute instance can be rerun.

Because the startup script has the conditional functionality, it determines that Yugabyte is installed, that the configuration files are available, and now launches Yugabyte master and tserver. This is the exact same situation as the compute instance restarting.

Overall installation process

To summarize, here the overall installation process step-by-step:

1. **Create compute instances.** Create compute instances (all 37, one in each Google Cloud region)
2. **Execute startup script automatically.** The creation of a compute instance executes the startup script automatically. Since this is a new compute instance the startup installs or updates the dependencies and then installs Yugabyte itself. Since no configuration files are available, the startup script does not try to launch Yugabyte.
3. **Create configuration files.** After all compute instances are created and running, run the script that creates the configuration files for Yugabyte master and tserver and that stores them on each compute instance. This script retrieves the internal IP addresses of the compute instances before creating the configuration files.
4. **Rerun startup scripts.** Iterate through all compute instances and rerun the startup script. The startup script detects that Yugabyte is installed and the configuration files are present. As a consequence it is launching the Yugabyte master and tserver on each compute instance.
5. **Wait for operational Yugabyte cluster.** Eventually the Yugabyte cluster with 37 nodes is running and operational.

Example restart of compute instance

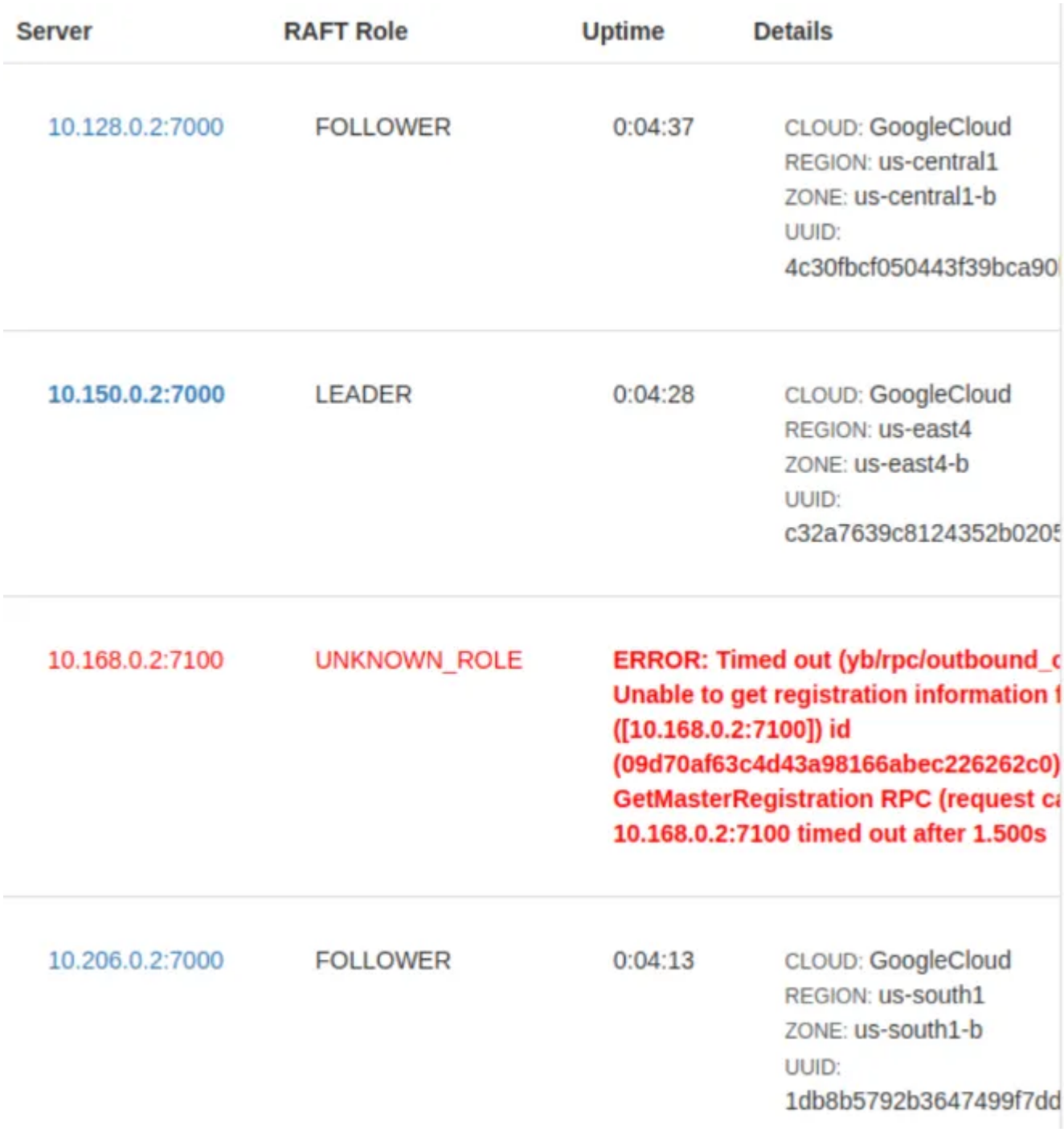
The following shows an example restart of a compute instance. Of course, the most interesting instance to restart is that of the leader node in a Yugabyte cluster. In the example the leader is initially in `us-west2`:

Server	RAFT Role	Uptime	Details
10.128.0.2:7000	FOLLOWER	0:01:37	CLOUD: GoogleCloud REGION: us-central1 ZONE: us-central1-b UUID: 4c30fbcf050443f39bca9(
10.150.0.2:7000	FOLLOWER	0:01:27	CLOUD: GoogleCloud REGION: us-east4 ZONE: us-east4-b UUID: c32a7639c8124352b02(
10.168.0.2:7000	LEADER	0:01:21	CLOUD: GoogleCloud REGION: us-west2 ZONE: us-west2-b UUID: 09d70af63c4d43a98166
10.206.0.2:7000	FOLLOWER	0:01:13	CLOUD: GoogleCloud REGION: us-south1 ZONE: us-south1-b UUID: 1db8b5792b3647499f7d

Steampipe shows the status of the compute instances (a small Yugabyte cluster with 4 nodes only for the purpose of this demonstration):

```
> select name, status from gcp.gcp_compute_instance;
+--------------------------+---------+
| name                     | status  |
+--------------------------+---------+
| yb-instance-us-central1 | RUNNING |
| yb-instance-us-south1   | RUNNING |
| yb-instance-us-east4    | RUNNING |
| yb-instance-us-west2    | RUNNING |
+--------------------------+---------+
```

When shutting down the compute instance that is running the leader, a new leader gets automatically chosen, now in `us-east4`:



Server	RAFT Role	Uptime	Details
10.128.0.2:7000	FOLLOWER	0:04:37	CLOUD: GoogleCloud REGION: us-central1 ZONE: us-central1-b UUID: 4c30fbcf050443f39bca90
10.150.0.2:7000	LEADER	0:04:28	CLOUD: GoogleCloud REGION: us-east4 ZONE: us-east4-b UUID: c32a7639c8124352b020
10.168.0.2:7100	UNKNOWN_ROLE	**ERROR: Timed out (yb/rpc/outbound_c Unable to get registration information f ([10.168.0.2:7100]) id (09d70af63c4d43a98166abec226262c0) GetMasterRegistration RPC (request ca 10.168.0.2:7100 timed out after 1.500s**	
10.206.0.2:7000	FOLLOWER	0:04:13	CLOUD: GoogleCloud REGION: us-south1 ZONE: us-south1-b UUID: 1db8b5792b3647499f7dd

Steampipe confirms that the former leader's compute instance in `us-west2` is stopped:

```
> select name, status from gcp.gcp_compute_instance;
+--------------------------+------------+
| name                     | status     |
+--------------------------+------------+
| yb-instance-us-east4     | RUNNING    |
| yb-instance-us-south1    | RUNNING    |
| yb-instance-us-central1  | RUNNING    |
| yb-instance-us-west2     | TERMINATED |
+--------------------------+------------+
```

Once `us-west2` is up and running again, Yugabyte was restarted and the Yugabyte node now is a follower:

Server	RAFT Role	Uptime	Details
10.128.0.2:7000	FOLLOWER	0:08:02	CLOUD: GoogleCloud REGION: us-central1 ZONE: us-central1-b UUID: 4c30fbcf050443f39bca90b32b4e
10.150.0.2:7000	LEADER	0:07:53	CLOUD: GoogleCloud REGION: us-east4 ZONE: us-east4-b UUID: c32a7639c8124352b0205efbaal
10.168.0.2:7000	FOLLOWER	0:00:04	CLOUD: GoogleCloud REGION: us-west2 ZONE: us-west2-b UUID: 09d70af63c4d43a98166abec22(
10.206.0.2:7000	FOLLOWER	0:07:38	CLOUD: GoogleCloud REGION: us-south1 ZONE: us-south1-b UUID: 1db8b5792b3647499f7dd976a0

A steampipe query confirms that all compute instances are running again:

```
> select name, status from gcp.gcp_compute_instance;
+--------------------------+---------+
| name                     | status  |
+--------------------------+---------+
| yb-instance-us-west2     | RUNNING |
| yb-instance-us-central1  | RUNNING |
| yb-instance-us-east4     | RUNNING |
| yb-instance-us-south1    | RUNNING |
+--------------------------+---------+
```

Limits and benefits of startup script approach

One limitation of the startup script approach in combination with configuration files is that it only works when compute instances are restarted. It does not work when a compute instance is destroyed and recreated: in that case the configuration files for the recreated compute instance must be stored on the compute instance first. This is future work for me.

The key benefit compared to providing the configuration as command line parameters is that the restart of a compute instance starts Yugabyte automatically. If using only command lines, then a restart of compute instance would have to be followed by an external operational script that connects to the compute instance and starts up Yugabyte providing command line parameters.

A better canvas for future work

At this point I can create a global Yugabyte cluster with 37 nodes where each node can be restarted ensuring that the corresponding Yugabyte master and tserver are restarted as well. This provides nice resilience for compute instance restart behavior, independent of the reason for a restart.

While I could focus more on operational aspects first (like parallelization of configuration file creation, or recreation of compute instances), I am eager to dive into geo-partitioning for a global Yugabyte cluster. I'll report in future blogs about my progress.

Yugabyte | Resilience | Restart | Configuration | Global Database

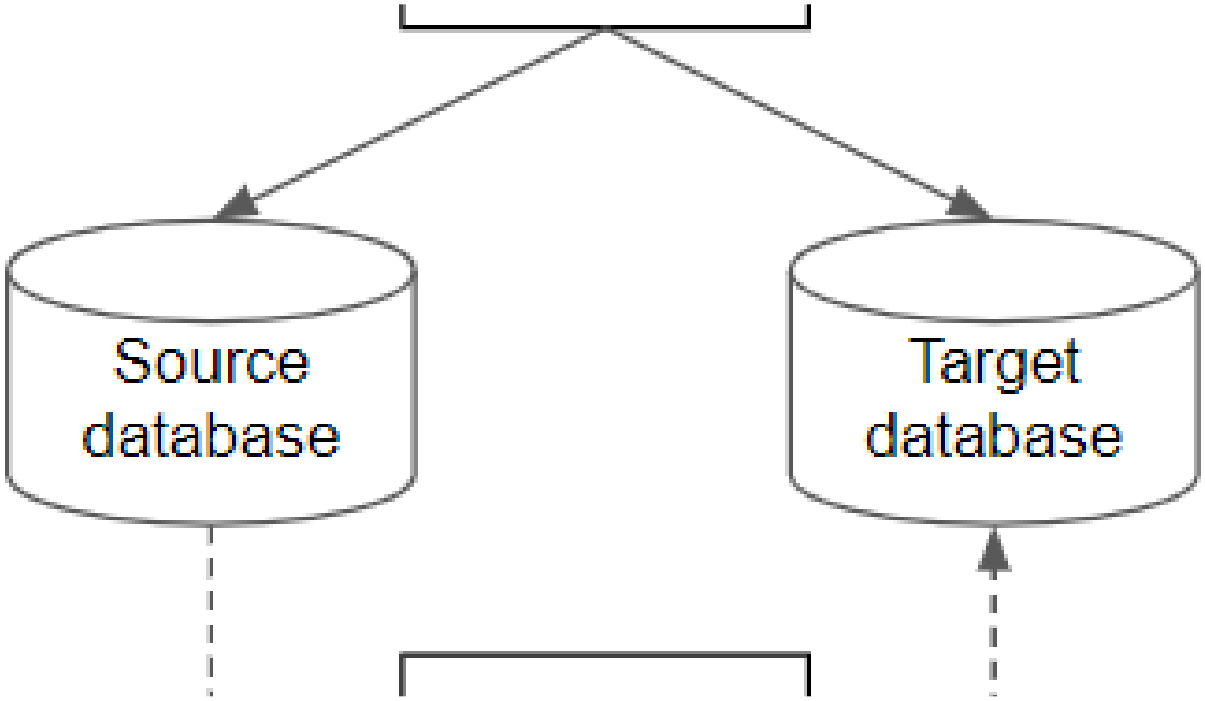
Edit profile

Written by Christoph Bussler

111 Followers

www.real-programmer.com

More from Christoph Bussler

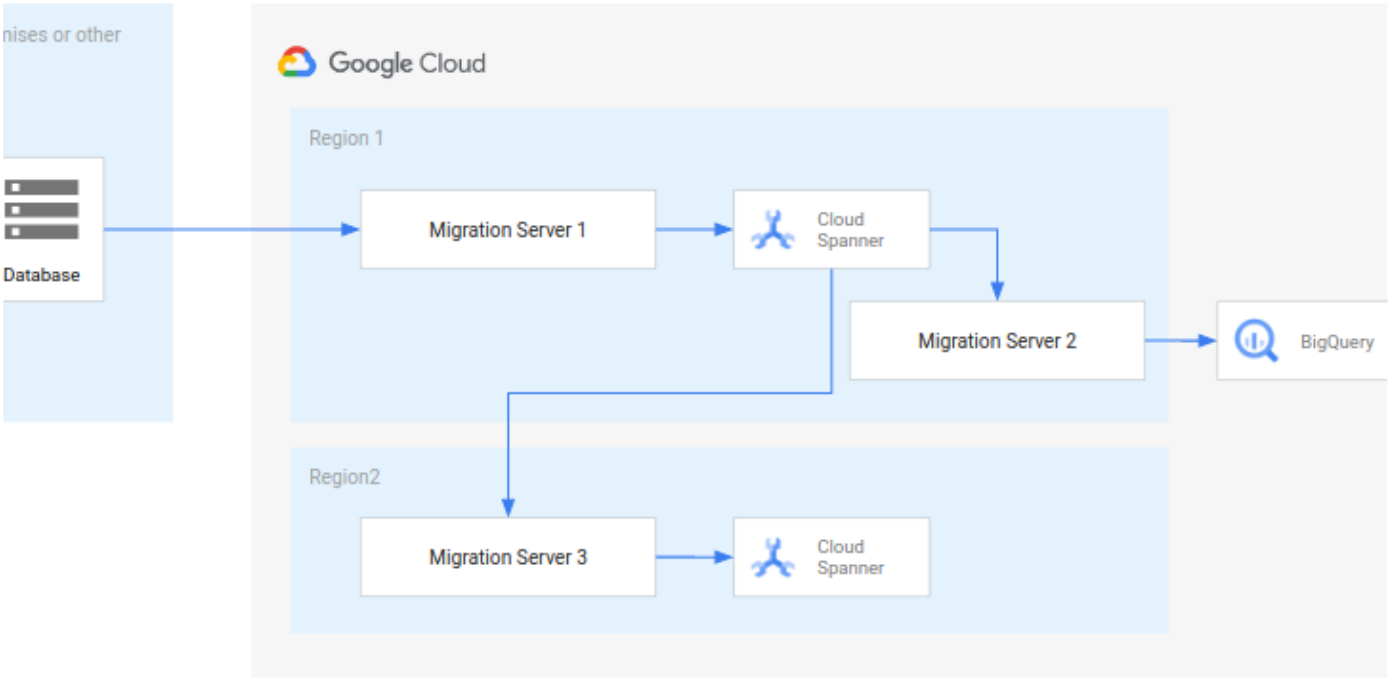


Christoph Bussler in Google Cloud - Community

Online Database Migration by Dual-Write: This is not for Everyone

(to be more precise: for almost no-one)

47 min read · Jun 23, 2020



Christoph Bussler in Google Cloud - Community

Zero downtime database migration and replication to and from Cloud Spanner

by Christoph Bussler, Szabolcs Rozsnyai

12 min read · Jul 13, 2020

ingers(1)	"Marc"	"Richards"	<Bytes>	
lbums(1, 1)				"Total Junk"
lbums(1, 2)				"Go, Go, Go"
ingers(2)	"Catalina"	"Smith"	<Bytes>	
lbums(2, 1)				"Green"
lbums(2, 2)				"Forever Hold Your Peac
lbums(2, 3)				"Terrified"

Christoph Bussler in Google Cloud - Community

Cloud Spanner's Table Interleaving: A Query Optimization Feature

By Christoph Bussler and Anand Jain

11 min read · Apr 16, 2021

26

Christoph Bussler in Google Cloud - Community

Database Migration and Replication: On Consistency

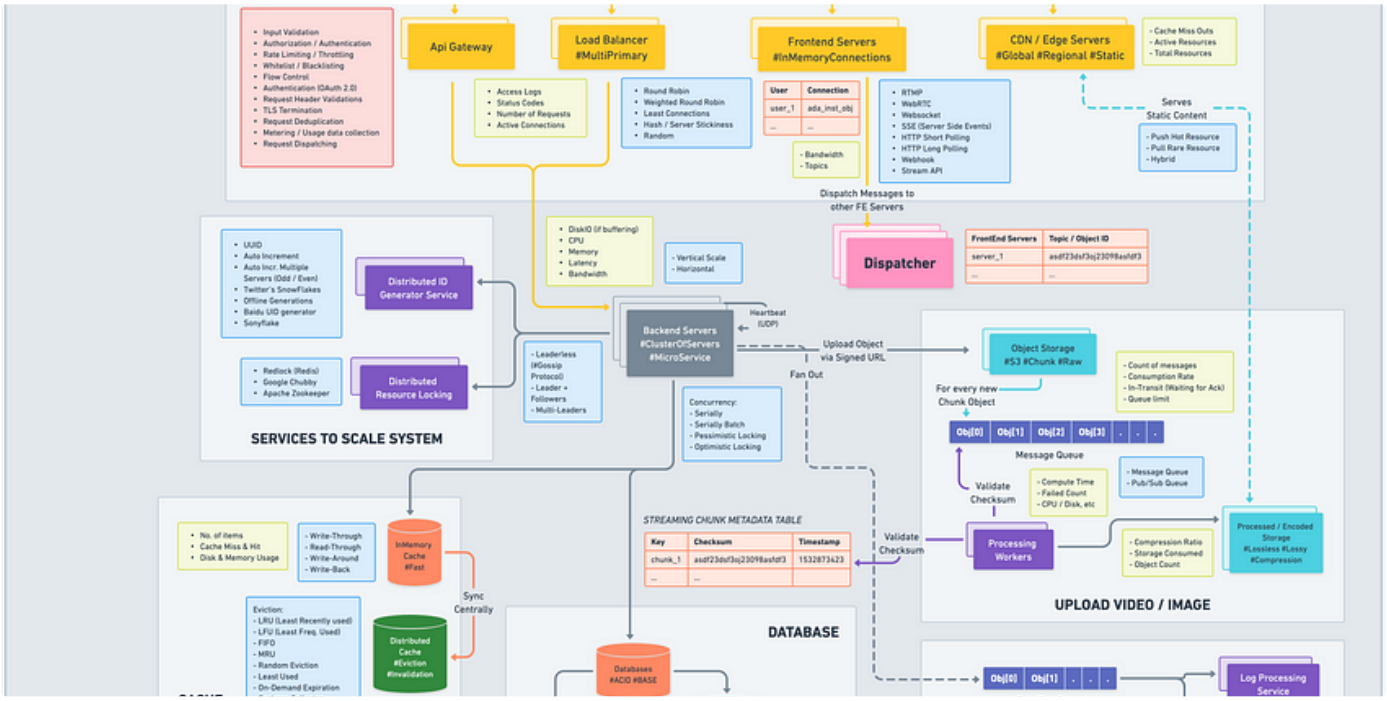
Application of key replication axioms ensure consistency

11 min read · Jun 11, 2021

71

See all from Christoph Bussler

Recommended from Medium



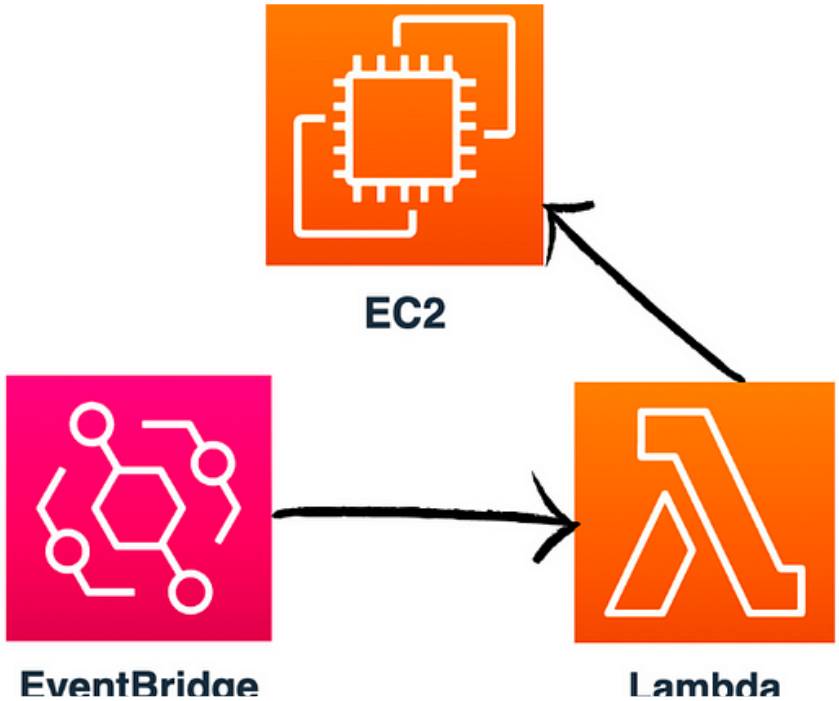
Love Sharma in Dev Genius

System Design Blueprint: The Ultimate Guide

Developing a robust, scalable, and efficient system can be daunting. However, understanding the key concepts and components can make the...

· 9 min read · Apr 20

4.4K 31



Prashant Gaur

Automating EC2 Instance Management with AWS Lambda and EventBridge

Introduction: Managing EC2 instances in AWS can be a repetitive and time-consuming task. Manually starting or stopping instances based on a...

3 min read · 6 days ago

Lists

Staff Picks

369 stories · 130 saves

Stories to Help You Level-Up at Work

19 stories · 124 saves

Self-Improvement 101

20 stories · 208 saves

Productivity 101

20 stories · 226 saves

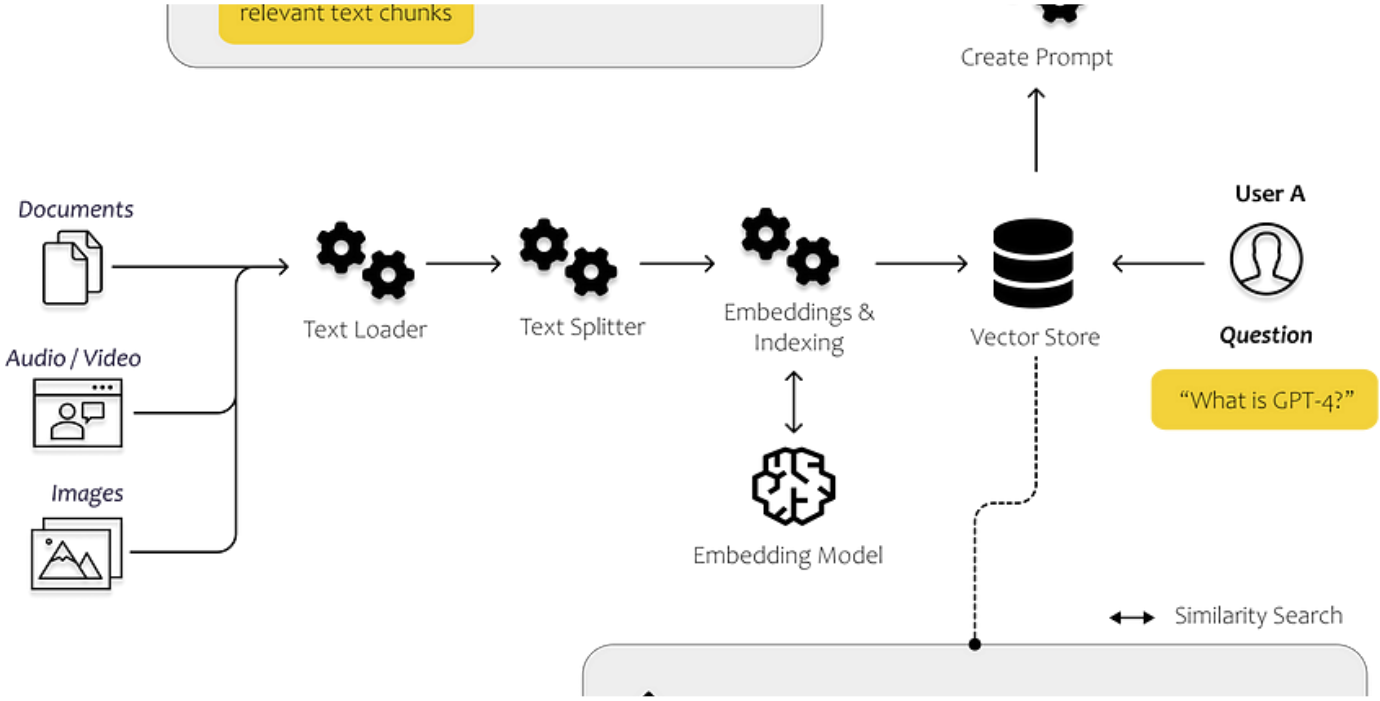
Nico Hein in Apache Airflow

Running a Multi-Tenant Airflow Cluster

Apache Airflow is a versatile and well maintained open source tool for orchestrating big data and analytics workloads. In this blog post...

7 min read · Jun 21

55



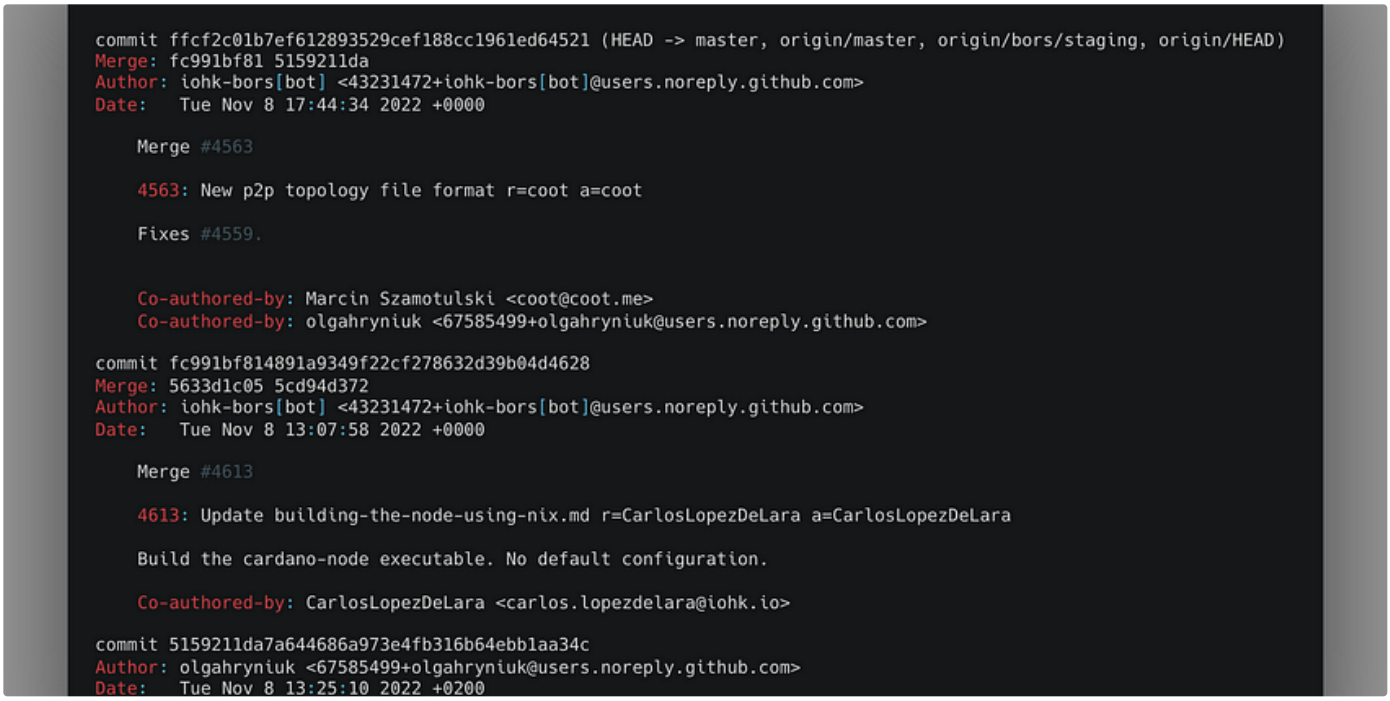
Dominik Polzer in Towards Data Science

All You Need to Know to Build Your First LLM App

A step-by-step tutorial to document loaders, embeddings, vector stores and prompt templates

✨ · 25 min read · 6 days ago

524 6



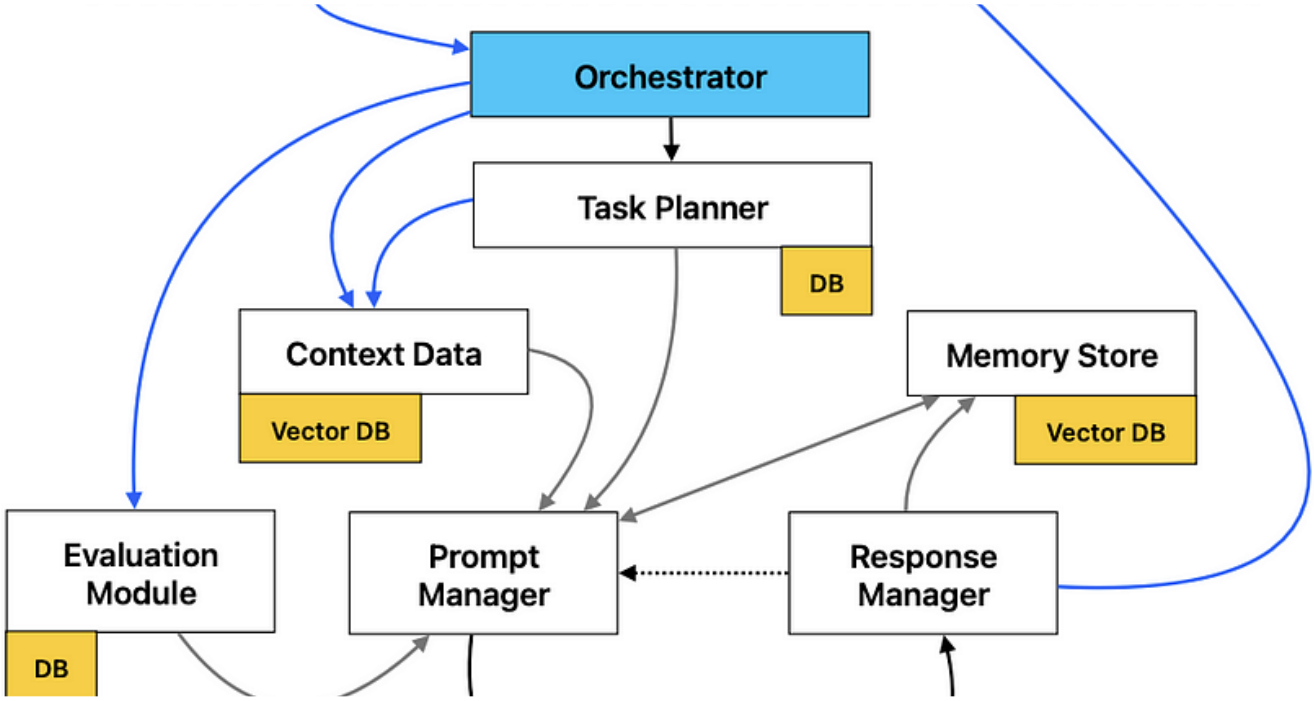
Jacob Bennett in Level Up Coding

Use Git like a senior engineer

Git is a powerful tool that feels great to use when you know how to use it.

· 4 min read · Nov 14, 2022

6.7K 67



Simon Attard

Leveraging Large Language Models in your Software Applications

How can you leverage the capabilities of Large Language Models (LLMs) within your software applications?

12 min read · Jun 20

438 5