



# Rollout Automation with Temporal

How to upgrade 1000+ ClickHouse Clusters

July / 2024

# Table of contents

**01**

## **ClickHouse Cloud**

What is “ClickHouse Cloud” and how does it work behind the scenes?

**02**

## **Upgrading ClickHouse in the Cloud**

What’s the challenge of upgrading ClickHouse clusters in the cloud?

**03**

## **Durable Execution & Temporal**

What is Temporal, how does it work and why is perfect for rollout automation?

**04**

## **Our solution**

Some examples how we do rollouts today (compared to one year ago)



01

# ClickHouse Cloud

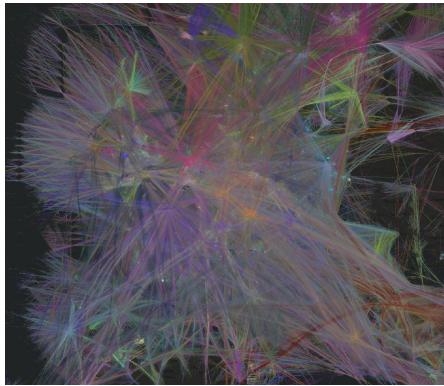
What is "ClickHouse Cloud" and how does it work behind the scenes?

# What is ClickHouse?

Open source	column-oriented	distributed	OLAP database
Developed since 2009 OSS 2016 35k+ Github stars 1k+ contributors 300+ releases	Best for aggregations Files per column Sorting and indexing Background merges	Replication Sharding Multi-master	Analytics use cases Aggregations Visualizations Mostly immutable data

# What is ClickHouse?

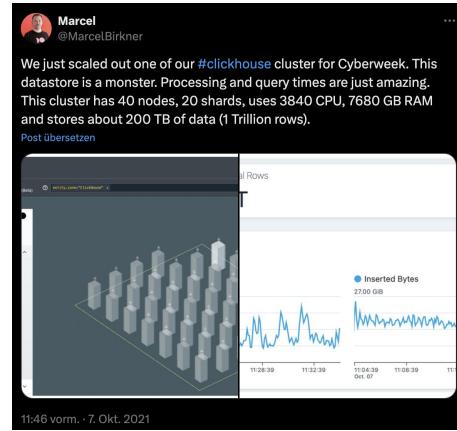
**adsb.exposed**



**clickpy.clickhouse.com**



## Instana

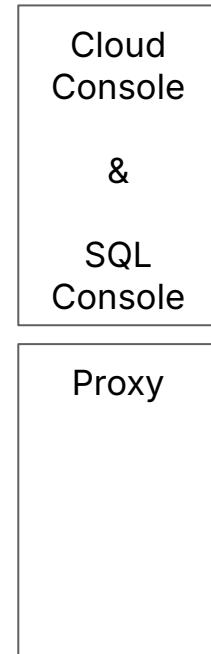
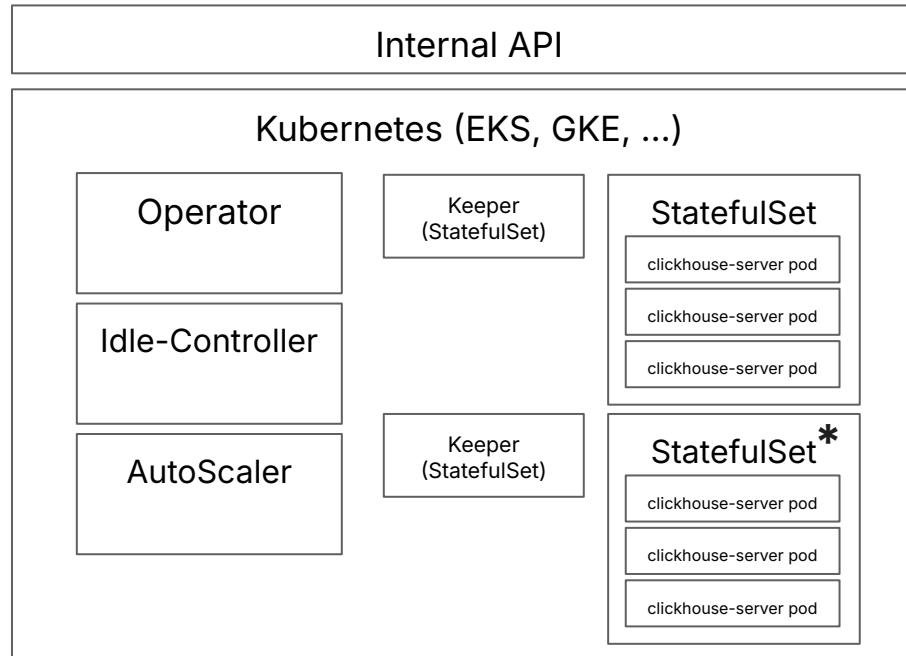


# What is special about ClickHouse “Cloud”?

- It's a **service** - sign up and start using it
- Separation of Storage and Compute
  - Unlimited Storage (S3, GCS, Azure BlobStorage)
  - Compute scales independently

# Architecture of ClickHouse Cloud

(quite complex)



\* We actually have one StatefulSet per Pod - see Manish's KubeCon talk  
["Fantastic Ordinals and How to Avoid Them: Auto-Scaling Challenges in a Cloud Database"](#)

# Architecture of ClickHouse Cloud

(simplified)

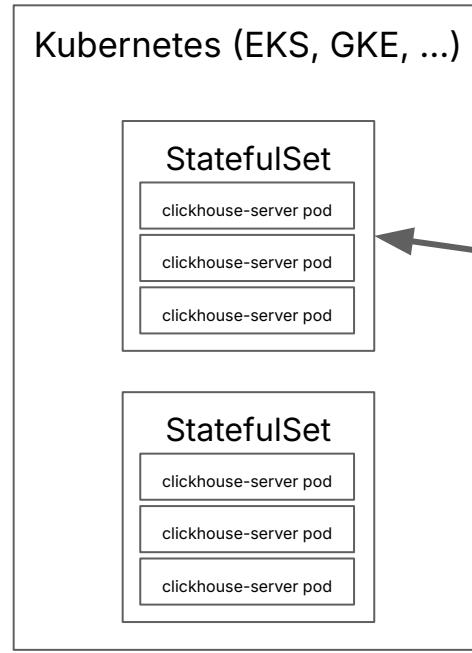


Amazon S3



Google Cloud Storage

Microsoft Azure  
Blob Storage





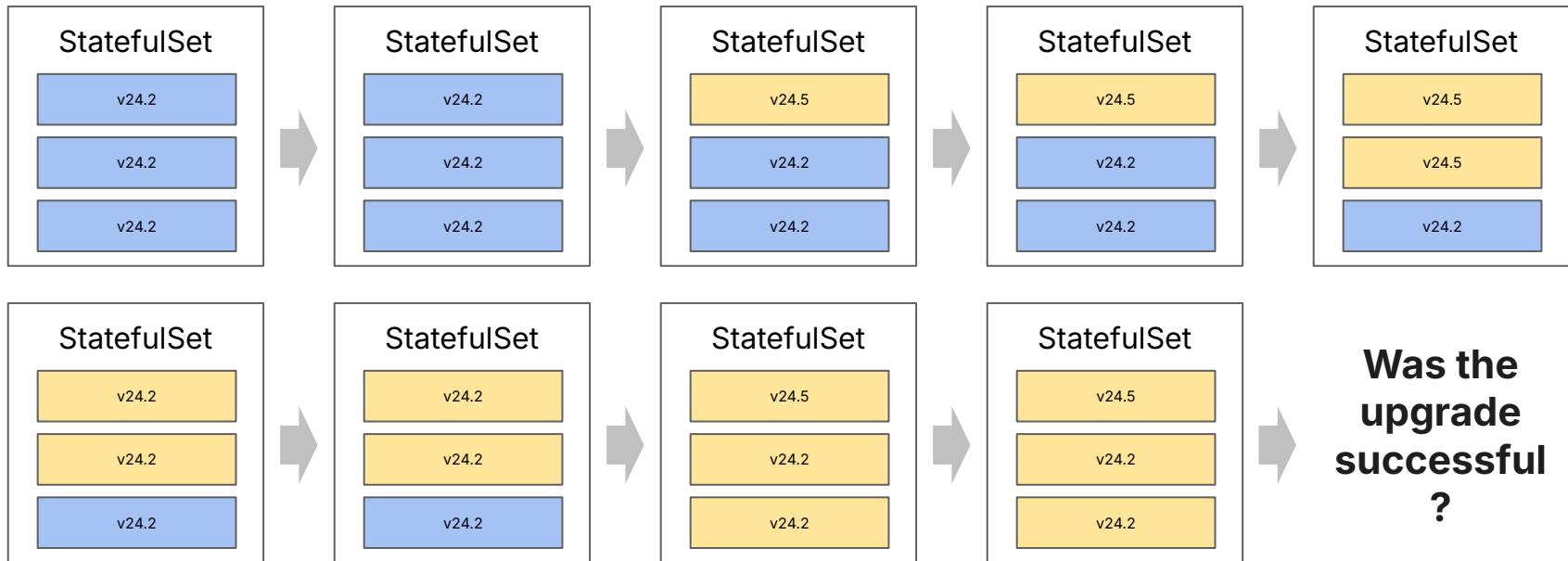
02

# Upgrading ClickHouse

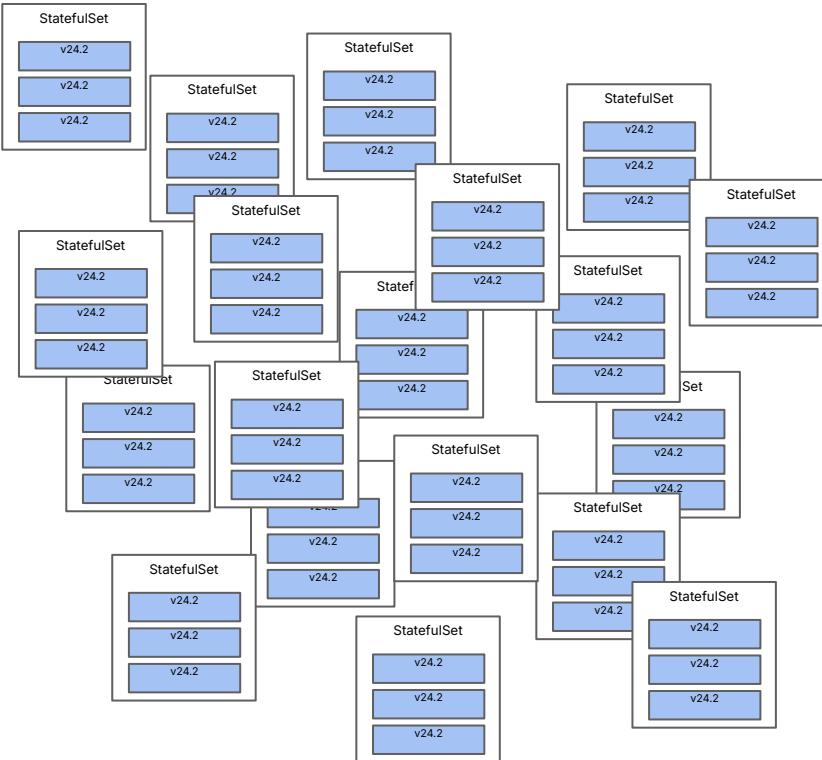
What's the challenge of upgrading ClickHouse clusters in the cloud?

# Upgrading a ClickHouse cluster

```
kubectl patch sts my-clickhouse -p \  
'{"spec": {"template": {"spec": {"containers": [{"name": "clickhouse", "tag": "24.5"}]} }}}'
```



# Upgrading 1000+ ClickHouse Clusters



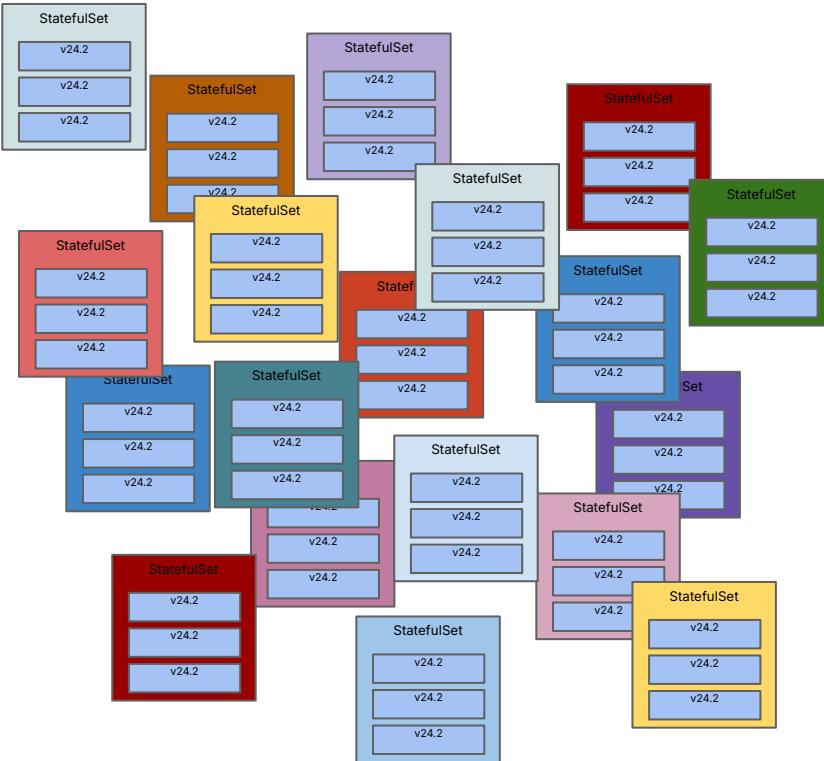
Naive Approach: all at once

- Will overload infrastructure
- Huge blast radius of potential bugs

Naive Approach #2: one after the other

- Too slow

# Upgrading 1000+ ClickHouse Clusters



- **Vastly different use cases** (a new version can be perfectly fine for 98% of the customers and break horribly for the 20 remaining clusters)
- Slightly different configurations and feature flags
- Differences across cloud providers
- ...

# Upgrading 1000+ ClickHouse Clusters

Requirements for a rollout automation system

- As fast as possible, but slow enough to still give time to react
- Runs unattended for several days
- Ability to inspect progress
- Allows interaction with engineers (e.g. pause & resume)
- Can integrate with other systems and tools  
(e2e tests, monitoring, slack, github)
- Flexible

**03**

# Temporal

What is “Durable Execution” and why is perfect for rollout automation?

# How hard can it be ...

- to call an unreliable API reliably
- wait for up to 3h for a result
- repeat these steps 1000 times
- **even if the process is restarted a dozen times in between**

# What is “durable execution”?

<https://temporal.io/blog/building-reliable-distributed-systems-in-node>

Durable execution systems run our code in a way that **persists each step** the code takes. If the process or container running the code dies, the code automatically continues running in another process with all state intact, including call stack and local variables.

# Temporal Concepts

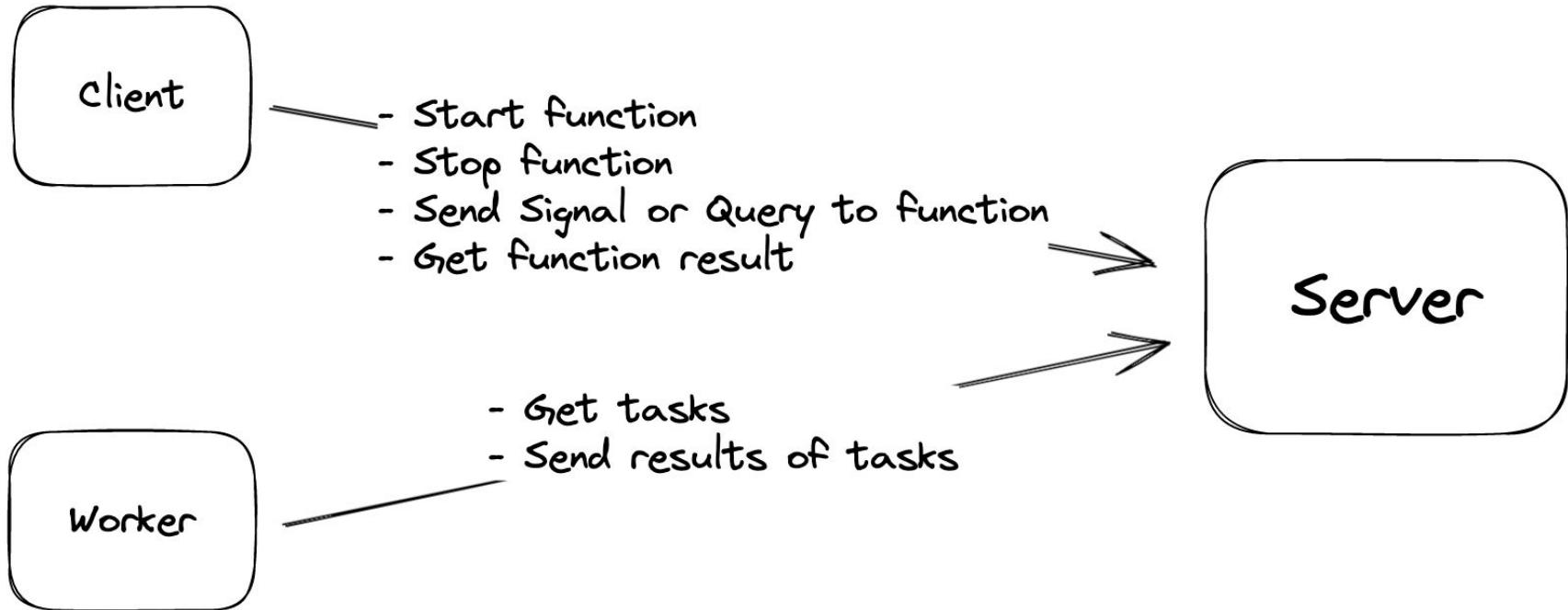
## Workflow

A Temporal Workflow defines the overall flow of the application. Conceptually, a Workflow is a sequence of steps written in a general-purpose programming language.

## Activity

An Activity is an implementation of a task to be performed as part of a larger Workflow. [...] It is in the Activities where all operations that affect the desired results must be implemented.

# How does “durable execution” work?



# Demo

## Demo 1 - Workflow continues after worker interrupted

temporal (temporal)  
temporal server start-dev > /tmp/temporal-dev-server-log

go (main)

```
+ munich-sre-meetup git:(step-1-simple-workflow-with-sleep) go run worker/main.go | jq -c '{time, msg}'
{"time": "2024-08-05T17:21:13.099552+02:00", "msg": "Started Worker"}
```

```
{"time": "2024-08-05T17:21:17.534551+02:00", "msg": "HelloWorld workflow started"}
```

```
{"time": "2024-08-05T17:21:17.534559+02:00", "msg": "It's now 2024-08-05 15:21:17.532057 +0000 UTC"}
```

```
{"time": "2024-08-05T17:21:17.534564+02:00", "msg": "Sleeping until 2024-08-05 15:21:47.532057 +0000 UTC"}
```

```
{"time": "2024-08-05T17:21:17.534609+02:00", "msg": "NewTimer"}
```

```
AC
```

```
+> munich-sre-meetup git:(step-1-simple-workflow-with-sleep) echo "worker down..."
```

```
worker down...
```

```
+> munich-sre-meetup git:(step-1-simple-workflow-with-sleep) go run worker/main.go | jq -c '{time, msg}'
{"time": "2024-08-05T17:21:38.428675+02:00", "msg": "Started Worker"}
```

```
{"time": "2024-08-05T17:21:37.543168+02:00", "msg": "Done with sleeping. It's now 2024-08-05 15:21:37.540191 +0000 UTC"}
```

... .ch-sre-meetup (-zsh)

munich-sre-meetup git:(step-1-simple-workflow-with-sleep) go run starter/main.go

```
2024/08/05 17:21:17 INFO: No logger configured for temporal client. Created default one.
```

```
2024/08/05 17:21:17 Started workflow WorkflowID hello_world_workflowID RunID 3fed9445-41c3-45d7-973b-f60bc4319304
```

```
2024/08/05 17:21:37 Workflow result: workflow-result
```

munich-sre-meetup git:(step-1-simple-workflow-with-sleep) []

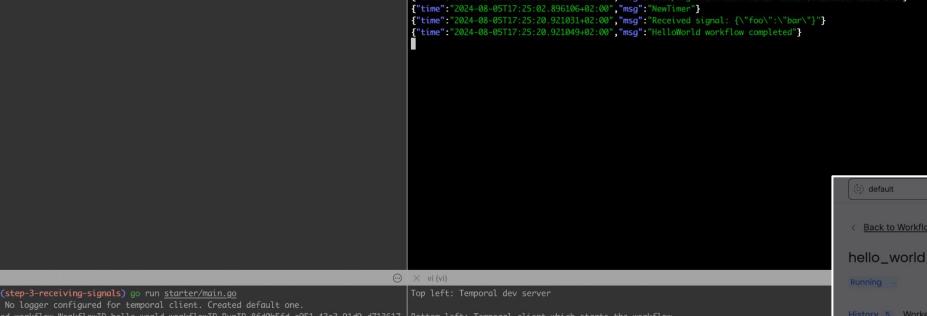
Top left: Temporal dev server  
Bottom left: Temporal client which starts the workflow  
Top right: The important part. The temporal worker which actually does the work. It is interrupted.  
But it continues work after starting up again

# Demo 2 - Temporal takes care of retries

The image shows a terminal session with four panes illustrating a workflow execution involving retries.

- Top Left:** A terminal window titled "temporal [temporal]" showing the command "temporal server start-dev > /tmp/temporal-dev-server-log".
- Top Right:** A terminal window titled "go (main)" showing the logs of a Temporal worker. The logs detail the execution of a "HelloWorld" workflow, including the creation of a worker, the execution of an activity, and multiple attempts to read a file /tmp/result until success.
- Bottom Left:** A terminal window titled "ch-sre-meetup (~zsh)" showing the command "git:(step-2-activity-with-retries) go run worker/main.go". The logs show the creation of a workflow and the execution of an activity, which prints "Workflow result: foo".
- Bottom Right:** A terminal window titled "x (vim)" showing the command "git:(step-2-activity-with-retries) go run starter/main.go". The logs show the creation of a workflow and the execution of an activity, which prints "Workflow result: foo". A note above this pane states: "Top left: Temporal dev server", "Bottom left: Temporal client which starts the workflow", and "Top right: The important part. The temporal worker which actually does the work. It tries to read a file /tmp/result - and retries until the file can actually be read."

## Demo 3 - Interacting with the workflow using signals



terminal (temporal)

```
temporal server start-dev > /tmp/temporal-dev-server-log
```

go (main)

```
munch-sre-meetup git:(step-3-receiving-signals) go run worker/main.go
{
  "time": "2024-08-05T17:25:01.573313+02:00", "msg": "Started Worker"
  {"time": "2024-08-05T17:25:02.895393+02:00", "msg": "HelloWorld workflow started"}
  {"time": "2024-08-05T17:25:02.896032+02:00", "msg": "It's now 2024-08-05 15:25:02.890006 +0000 UTC"}
  {"time": "2024-08-05T17:25:02.896092+02:00", "msg": "Sleeping until 2024-08-05 15:25:32.890006 +0000 UTC"}
  {"time": "2024-08-05T17:25:20.921051+02:00", "msg": "Received signal: (\\"foo\\",\\"bar\\")"}
  {"time": "2024-08-05T17:25:20.921069+02:00", "msg": "HelloWorld workflow completed"}
}
```

.ch-sre-meetup (~23h)

```
munch-sre-meetup git:(step-3-receiving-signals) go run starter/main.go
2024/08/05 17:24:52 INFO No logger configured for temporal client. Created default one.
2024/08/05 17:24:52 Started workflow WorkflowID hello_world_workflowID RunID 8f0db5fd-e951-43e3-91d9-d713617
2024/08/05 17:25:20 Workflow result: result
munch-sre-meetup git:(step-3-receiving-signals) []
```

Top left: Temporal dev server

Top right: Temporal client which starts the workflow

Bottom right: The important part. The temporal worker which actually does the work. It sleeps until it receives a signal (done via web ui in this case)

default

Back to Workflows

hello\_world\_workflowID

Running

History 5 Workers 4 Pending Activities 0 Sub

\* Summary

Workflow Type

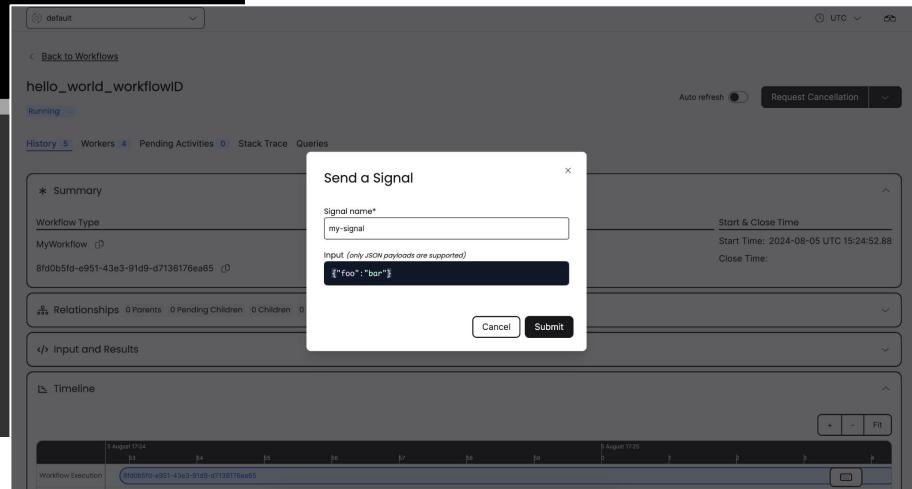
MyWorkflow

8f0db5fd-e951-43e3-91d9-d7136176ea65

Relationships 0 Parents 0 Pending Children

Input and Results

Timeline



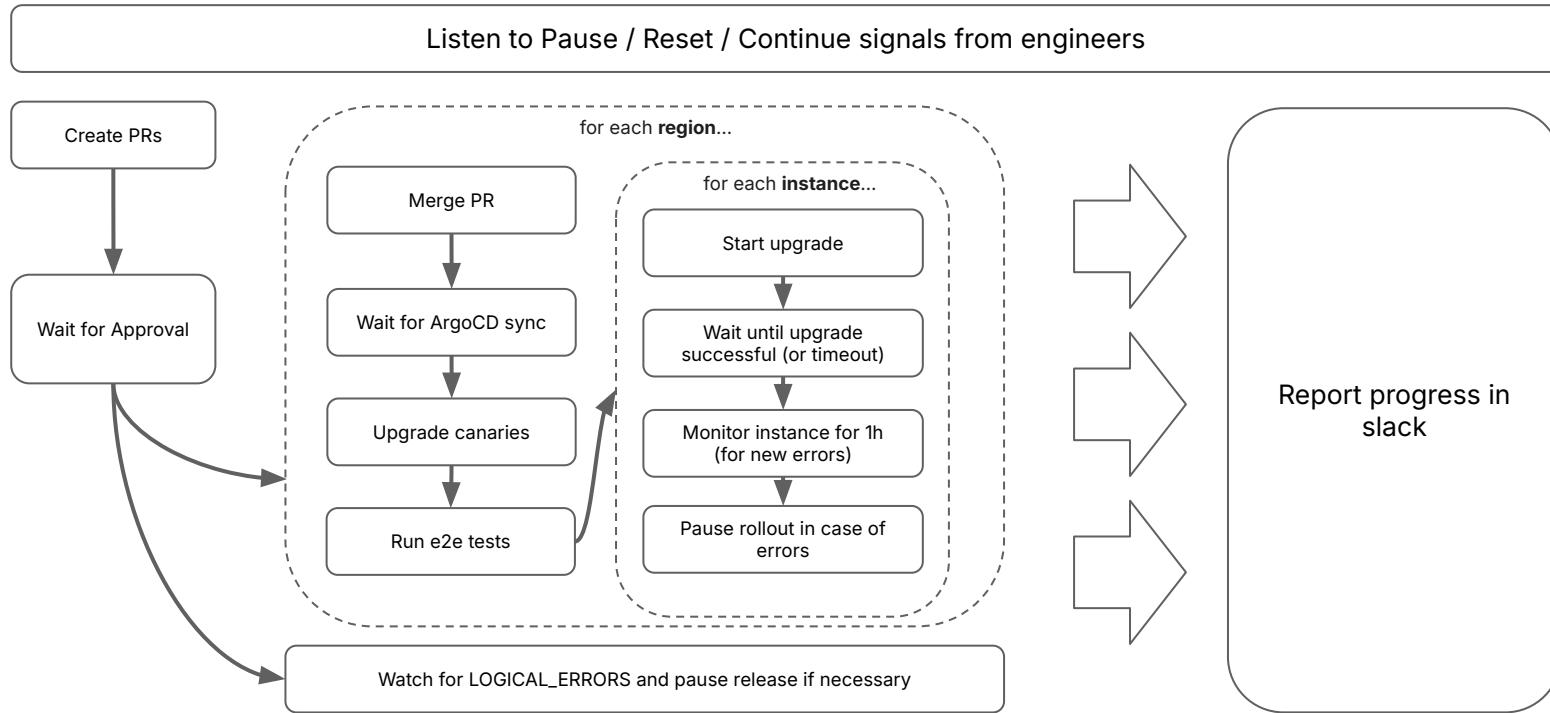
**04**

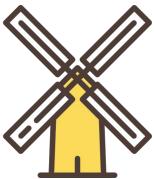
# Our solution

Some examples how we do rollouts today (compared to one year ago)



# ClickHouse Molen





# ClickHouse Molen



**Data-plane bot** APP 12:50  
[Main-distracted\\_northcutt](#)  
 Started new ClickHouse release: distracted\_northcutt  
 Release Engineer: [nikitamikhaylov@clickhouse.com](mailto:nikitamikhaylov@clickhouse.com)

Please review PRs  
 To start rollout send [Continue](#) signal to the main workflow. Send [Cancel](#) to cancel the rollout and cleanup PRs

**Release Schedule:**

- \* Production small regions
- \* Prod AWS eu-central-1, wait 1h0m0s
- \* Prod AWS eu-west-1, wait 1h0m0s
- \* Prod GCP europe-west4, wait 1h0m0s

```

- description: Update ClickHouse version
  operation: add_or_update
  path: clickhouse.version
  value: "23.9.2.47465"
- description: Change user-level setting
  operation: add_or_update
  path: extraConfig.usersprofiles.default.cluster_for_parallel_replicas
  value: "default"
  
```

47 replies Last reply today at 14:47

**Data-plane bot** APP 4 days ago  
[Release-adoring\\_turing](#)  
 Starting next stage 'Staging dev tier' in 0s at 16 Nov 23 15:17 UTC

**Data-plane bot** APP 4 days ago  
[Release-adoring\\_turing](#)  
 Starting stage: Staging dev tier

**Data-plane bot** APP 4 days ago  
[Group\\_rollout-adoring\\_turing-staging\\_dev\\_tier-staging\\_dev\\_tier](#)  
 Starting updating instances in group: Staging dev tier

**Data-plane bot** APP 4 days ago  
[Group\\_rollout-adoring\\_turing-staging\\_dev\\_tier-staging\\_dev\\_tier](#)  
 Instance update failed instance tan-dn-76 in staging/aws/eu-west-1 failed to update with error: 'initialState: degraded, instance update marked as failed as it exceeded the autoUpdate\_ttl timeout (type: NonRetryableError, retryable: false)'

**Data-plane bot** APP 4 days ago  
[Group\\_rollout-adoring\\_turing-staging\\_dev\\_tier-staging\\_dev\\_tier](#)  
 Instances in group Staging dev tier updated.

**Data-plane bot** APP 4 days ago  
[Release-adoring\\_turing](#)  
 Stage Staging dev tier is done\*Starting next stage 'Staging prod tier' in 0s at 16 Nov 23 15:59 UTC

**Data-plane bot** APP 4 days ago  
[Release-adoring\\_turing](#)  
 Starting stage: Staging prod tier

**Data-plane bot** APP 4 days ago  
[E2E-distracted\\_northcutt-aws-production-eu-central-1](#)  
 E2E tests finished for: aws/production/eu-central-1Seed: 2652460134  
 Success: false  
 To check failed specs follow [this link](#).  
 Logs: logs in DD

**Data-plane bot** APP 4 days ago  
[Update\\_base\\_configurations-distracted\\_northcutt-prod\\_aws\\_eu\\_central\\_1-prod\\_aws\\_eu\\_central\\_1](#)  
 Child workflow [E2E-distracted\\_northcutt-aws-production-eu-central-1](#) failed.\*To ignore this failure and continue release send [Continue](#) signal to [this workflow](#), otherwise send [Cancel](#) signal.

**Data-plane bot** APP 4 days ago  
[Errors\\_checks-distracted\\_northcutt-prod\\_aws\\_eu\\_central\\_1](#)  
 \*Logical errors detected\*Found logical errors in Cloud: aws Environment: production Region: eu-central-1 Cluster type: data-plane Cluster name: prod-production-eu-central-1-data-plane Namespace: ns-bisque-wa-97

**Data-plane bot** APP 2 minutes ago  
[Group\\_rollout-adoring\\_turing-staging\\_prod\\_tier-staging\\_prod\\_tier](#)  
 Rollout paused for all groups in stageStage will be paused because error rate 12 is over the limit 10. Send 'Continue' or 'Reset' to resume the rollout

**Data-plane bot** APP 3 days ago  
[tada Main-zealous\\_mclean](#)  
 ClickHouse release zealous\_mclean is done

# Temporal

## Great

- It's code  
testing, tooling, CI 🎉
- Active development  
test SDK, replay testing didn't exist when we started
- Flexible  
recently added better error monitoring and soft/hard timeouts

## Challenging

- Workflow versioning  
"nondeterministic workflow definition code or incompatible change"
- Limits  
Actions per Workflow, Child workflows, "ContinueAsNew", Size of Activity Input/Output
- Things work different than in normal go code - You need to embrace the paradigm!