Nightmare Infrastructures episode 2 – beSharp's Halloween special

27 October 2023 - 7 min. read

Boys and girls of every age

Wouldn't you like to see something strange?

Come with us, and you will see

This, our town of Halloween - The Nightmare before Christmas.

Last year, we saw some scary infrastructures. Are you ready for the new episode?

In this article, we'll see some strange infrastructure designs and practices we have encountered, telling stories about **Cloud anti-patterns that will become an absolute nightmare in the long term**.

Hold your breath; we're about to start!

The undead

The modern concept of zombies is influenced by the Haitian Vodou religion, where some people believe that a witch doctor can revive a dead person as their slave using magic or a secret potion.

Sometimes, ECS tasks are resurrected by a service, even if they should be buried (and stopped).

We saw it happen in a development environment when a small, pretty php container had a problem due to an error in a failed pipeline.

The task struggled to stay alive, but the brutal Application Load Balancer health check shot the container in the head; the ECS service did its best to revive the task, pulling it from the ECR repository.

No one noticed the issue until the end of the month when the billing increased to over 800$ due to 16 Terabytes of traffic through the NAT Gateway.

In this case, the ECS circuit breaker was willing to help, but no one asked him.

To avoid making ECS zombies, please involve him next time you deploy a container!

It’s a matter of trust

I’m lying when I say, "Trust me"
I can’t believe this is true…
Trust hurts, why does trust equal suffering?”
Megadeth, Trust

This is a short but even scarier episode. While investigating a problem with a failed pipeline deployment, we saw this trust policy in a role with administrator permissions on every resource:

```
{

  "Version": "2012-10-17",

  "Statement": [

    {

      "Effect": "Allow",
      "Principal": {

        "AWS": "*"

      },

      "Action": "sts:AssumeRole"

    }

  ]

}
```

We first detached the policy and, as a proof of concept, during a call with the customer, we used our personal AWS account to assume that role. Pretty scary, uh?

I see dead lambdas

Like in “The Sixth Sense”…

It was a cold winter night, and, during a storm, our on-duty cell phone started ringing desperately. A serverless application struggled to survive, and our API gateway desperately gave 5xx errors.

Our fellow colleague started investigating, and strangely, everything was quiet. Too quiet. No logs for the lambda associated with the troubled application route were recorded in CloudWatch.

When making requests with Postman or curl, everything worked like a charm.

Since everything was working again, the investigation was postponed until the next morning, but... After an hour, the phone started ringing again. And, still, no traces of failures, even in the logs.

Determined to solve the mystery, another colleague joined the investigation and, while reviewing the configuration... It suddenly appeared!

Our customer, in the past, was having some trouble because the lambda was timing out, so it “reserved some capacity”. It turned out that the “reserved concurrency” was set to 1.

According to the AWS documentation: "*Reserved concurrency is the maximum number of concurrent instances you want to allocate to your function. When a function has reserved concurrency, no other function can use that concurrency*".

But there's a catch: reserved concurrency is also the maximum number of concurrent lambda instances that can be executed, so setting this value to one effectively throttles and limits the lambda, so if two simultaneous users call the API route, API Gateway will return a 5xx error.

After removing the concurrency, everything was working fine. Consider using provisioned concurrency if you want to have lambdas ready to serve requests. This article explains how these two parameters impact the execution and performance Lambda function scaling - AWS Lambda (amazon.com).

Too many secrets

THERE ARE SOME SECRETS WHICH NOT NCT PERMITT THIEAIS ES BE TOLD.

"There are some secrets which do not permit themselves to be told"
- Edgar Allan Poe - The Man of the Crowd

Sometimes, infrastructures can be perfect, but they still can be abused.

This is the case of an e-commerce application replatforming gone berserk.

After an initial assessment and design, we suggested modifications to better integrate in a cloud environment.

Stateless sessions were implemented, thanks to Elasticache for Redis, Database scalability using Aurora Serverless for MySQL, and, finally, some application security thanks to the usage of roles and SecretsManager to store database credentials and external API keys instead of plain text config files.

Everything was fine when, after a deployment, the application was slowing down and failing load balancer health checks.

It turned out that since SecretsManager was so easy and handy to use, it was used for everything, even ordinary application parameters like bucket names, endpoints, and configuration thresholds. Every request to a page was accessing at least one or two times different secrets, and, to worsen things, there were traffic spikes due to marketing campaigns. At the end of the month, the AWS billing dashboard recorded 25,000,000 API calls, resulting in 150 $ added to the running costs.

But there's more: sometimes, the application stopped responding because the metadata service (used to authenticate API calls using IAM roles) throttled the access. After all, it thought the application was trying to make a DoS.

After explaining the issue and proposing a solution involving parameter store and environment variables, everything worked like a charm again, and the porting went fine.

High Unavailability

"Congratulations. You are still alive. Most people are so ungrateful to be alive. But not you. Not anymore" - Saw.

Speaking of application cloud-oriented refactoring of applications, this is a simple mistake that we sometimes see happen: if an application is highly available, please don't tie vital requests to a single point of failure endpoint (like an FTP server running on an EC2 instance to retrieve configuration files).

This is a short story, but there's a lot of suffering behind it.

Infernal backup

"We all go a little mad sometimes."
– Psycho (1960)

Disaster recovery of the on-premise workloads on the Cloud is a hot topic, especially for enterprises and on-premises architectures.

Designing a resilient solution is not easy. The first step is to find a backup strategy that can fit and guarantee the right RPO and RTO.

Once the backup strategy and requirements are defined, it's time to find the right software that fits our needs.

In a large enterprise, the first choice is to use the existing solution but adapt it to run into the Cloud. No problem, as long as there is at least a form of integration, typically with Amazon Glacier or Amazon S3. This was our case, but...

To make things more resilient, someone installed the software on an EC2 instance in a dedicated AWS account and configured it to backup on-premise machines using the existing Direct Connect connection through the Transit Gateway attachment.

You already can tell in which direction the AWS billing can go! To make things worse, all endpoints were centralized in a dedicated networking account for better manageability and observability.

So, to summarize, a single gigabyte backed up from the on-premise was:

- Using the Direct Connect connection
- Traversing the transit gateway attachment to reach the EC instance in the Backup Account
- Traversing the Transit Gateway to reach the network account
- Traversing the centralized S3 Interface endpoint

At the end of the month, a spike of 6,000 $ was in the Networking section of the AWS Bill.

Conclusion

What have we learned? If something seems easy and trivial in a complex environment, you should look better for cues.

Even this year, we had our fair dose of scary horror stories. As always, every mistake is not intentional, and the good thing is that we can always learn new things, avoiding repeating ourselves in the future.

I want to thank everyone at the AWS Community Day in Rome who shared their stories after my speech about last year's article. Next year's episode is already written! :)

What kind of horror did you find in your AWS accounts? Let us know in the comments! (we also accept anonymous e-mails :D)

About Proud2beCloud

Proud2beCloud is a blog by beSharp, an Italian APN Premier Consulting Partner expert in designing, implementing, and managing complex Cloud infrastructures and advanced services on AWS. Before being writers, we are Cloud Experts working daily with AWS services since 2007. We are hungry readers, innovative builders, and gem-seekers. On Proud2beCloud, we regularly share our best AWS pro tips, configuration insights, in-depth news, tips&tricks, how-tos, and many other resources. Take part in the discussion!

Damiano Giorgi

Ex on-prem systems engineer, lazy and prone to automating boring tasks. In constant search of technological innovations and new exciting things to experience. And that's why I love Cloud Computing! At this moment, the only "hardware" I regularly dedicate myself to is that my bass; if you can't find me in the office or in the band room try at the pub or at some airport, then!