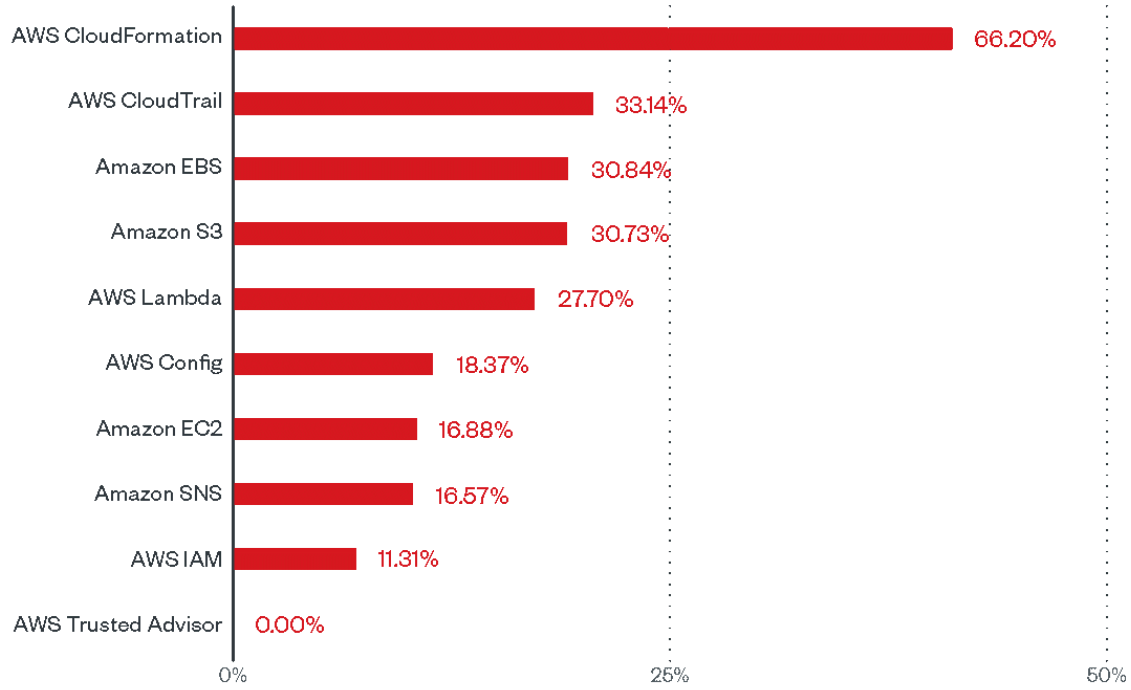
Securing **Infrastructure** in AWS

OWASP Kathmandu

Do we need to even worry about this?

- Research from McAfee's Cloud Adoption and Risk Report shows that on average, enterprises using IaaS/PaaS have 14 misconfigured services running at any given time, resulting in an average of 2,269 misconfiguration incidents per month!



Let me share an incident.

Background.

- Infrastructure is hybrid in nature (on prems. and AWS)
- It's a critical software with high value data in it.
- A downtime in this software could cost thousands of dollar to the business.

More on the Infrastructure.

- Dev / Staging / Prod were in separate AWS accounts.
- Staging and Prod (with on-prems) had some sort of shared network connectivity model.
- Completely managed via terraform.
- Multiple teams with multiple terraforms.
- **Team members were changing so frequently**, we had to automate IAM management.

And one day.

I got a message.

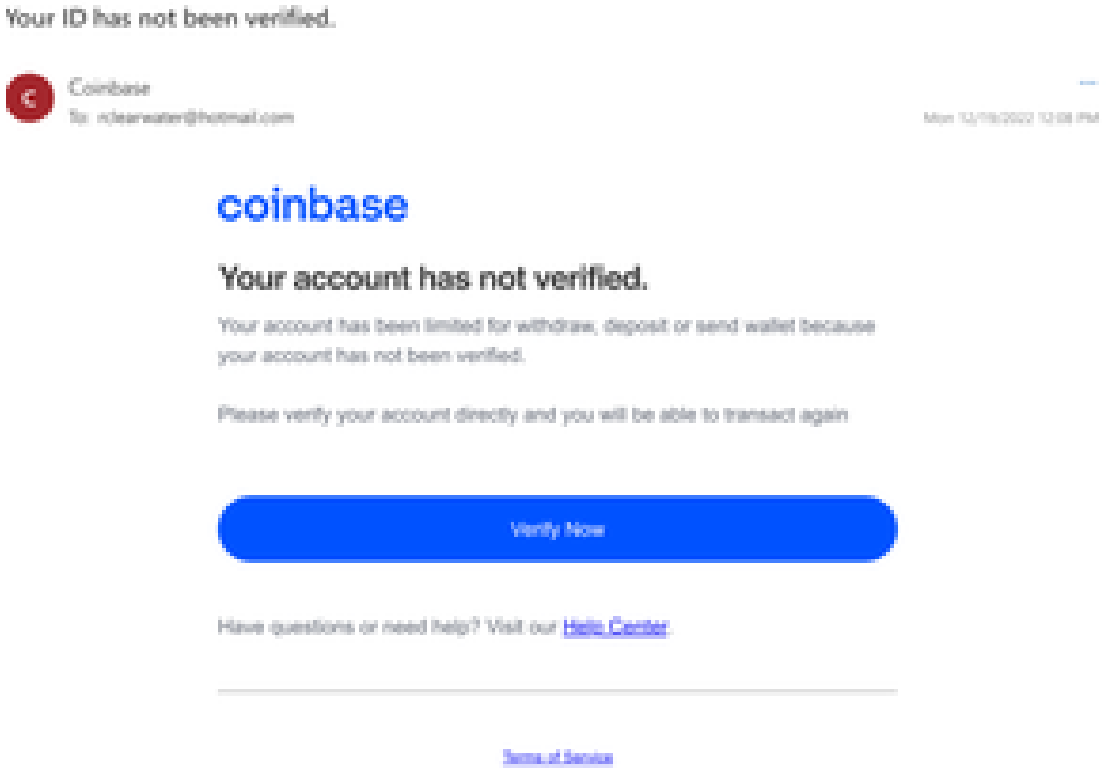
Mahesh, Dev ko SES ma kei problem aako ho? Mail gairako thyena so.

AWS account moved into under review.

57k mail sent in the span of 1-2hr.

Highly suspicious.

57k people received this **phishing** mail.



If you didn't guess already, our AWS account was breached.

How do we trace **who** did this?

Cloudtrail Logs

Username	Access Key	Timestamp	Service Url	Action	Region	IP	Used Software
[REDACTED]	[REDACTED]	2022-12-XT16:36:23Z	ses.amazonaws.com	GetSendQuota	us-east-1	[REDACTED]	aws-cli
[REDACTED]	[REDACTED]	2022-12-XT15:53:43Z	ses.amazonaws.com	GetSendQuota	us-east-1	[REDACTED]	aws-cli
[REDACTED]	[REDACTED]	2022-12-XT15:47:54Z	ses.amazonaws.com	GetSendQuota	us-east-1	[REDACTED]	aws-sdk-boto3
[REDACTED]	[REDACTED]	2022-12-XT15:47:54Z	ses.amazonaws.com	ListIdentities	us-east-1	[REDACTED]	aws-sdk-boto3
[REDACTED]	[REDACTED]	2022-12-XT15:47:28Z	ses.amazonaws.com	GetAccount	us-east-1	[REDACTED]	aws-cli-v2
[REDACTED]	[REDACTED]	2022-12-XT15:47:22Z	ses.amazonaws.com	GetSendQuota	us-east-1	[REDACTED]	aws-cli
[REDACTED]	[REDACTED]	2022-12-XT13:52:35Z	ses.amazonaws.com	ListIdentities	us-east-1	[REDACTED]	aws-sdk-boto3
[REDACTED]	[REDACTED]	2022-12-XT13:52:34Z	ses.amazonaws.com	GetAccount	us-east-1	[REDACTED]	aws-sdk-boto3
[REDACTED]	[REDACTED]	2022-12-XT13:10:10Z	ses.amazonaws.com	ListIdentities	us-east-1	[REDACTED]	aws-sdk-boto3
[REDACTED]	[REDACTED]	2022-12-XT13:10:10Z	ses.amazonaws.com	GetAccount	us-east-1	[REDACTED]	aws-sdk-boto3
[REDACTED]	[REDACTED]	2022-12-XT08:47:37Z	ses.amazonaws.com	ListIdentities	us-east-1	[REDACTED]	aws-sdk-go

What should be our **immediate** actions?

- Revoke IAM Permissions from the breached user with suspicious activity.
- Check into other environments / other services apart from AWS in which the user has access in.
- Check if AWS Support has something to say about the breach.
- Inform the user and ask about his activities on AWS.

How could we prevent this on first place?

- Strict AWS SSO implementation to authenticate physical users.
- Implement Least Privilege on users.
- Don’t enable production access to services on development environment.
- Authorize teams to only access abstraction layer for services impacting real users.

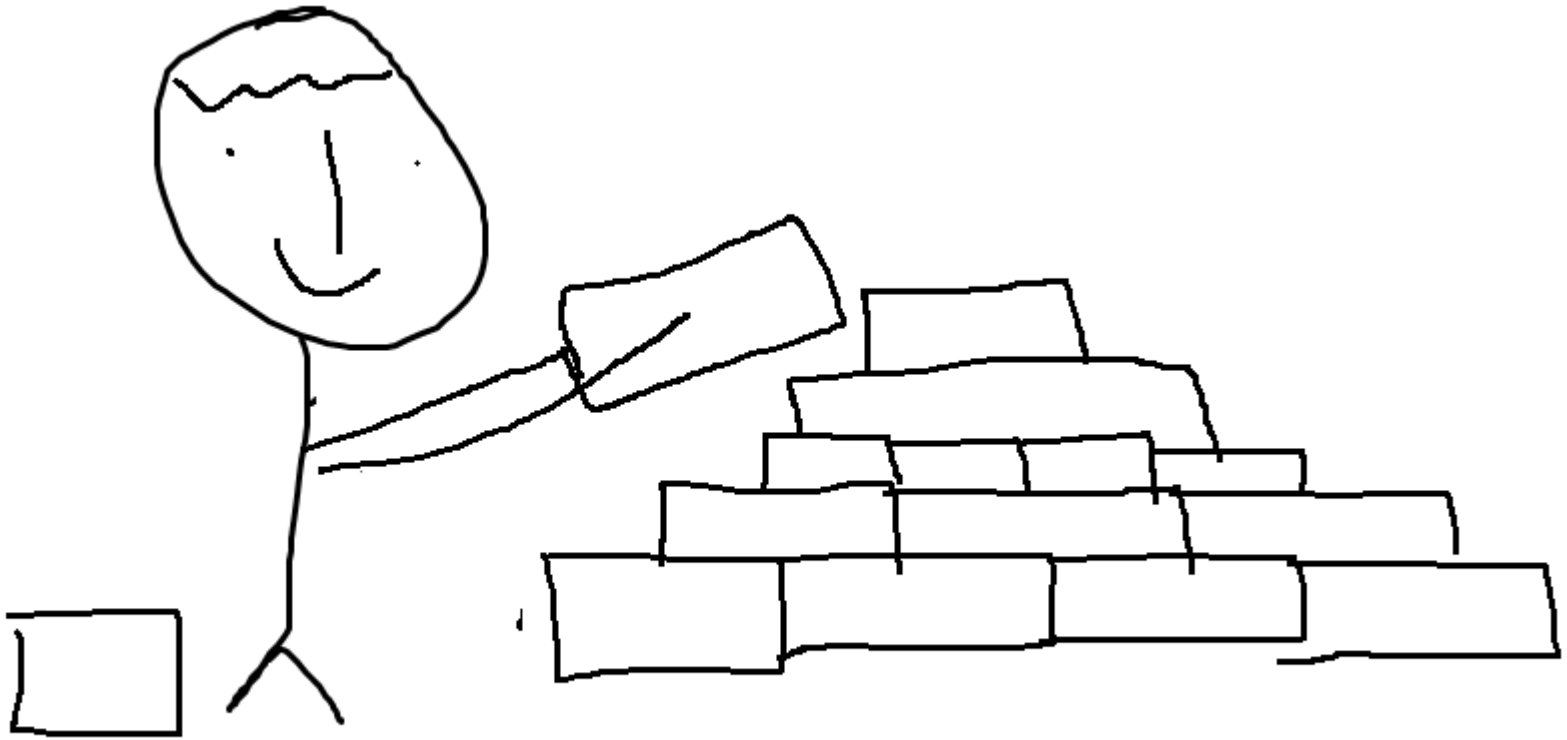
End of the story.

About Me

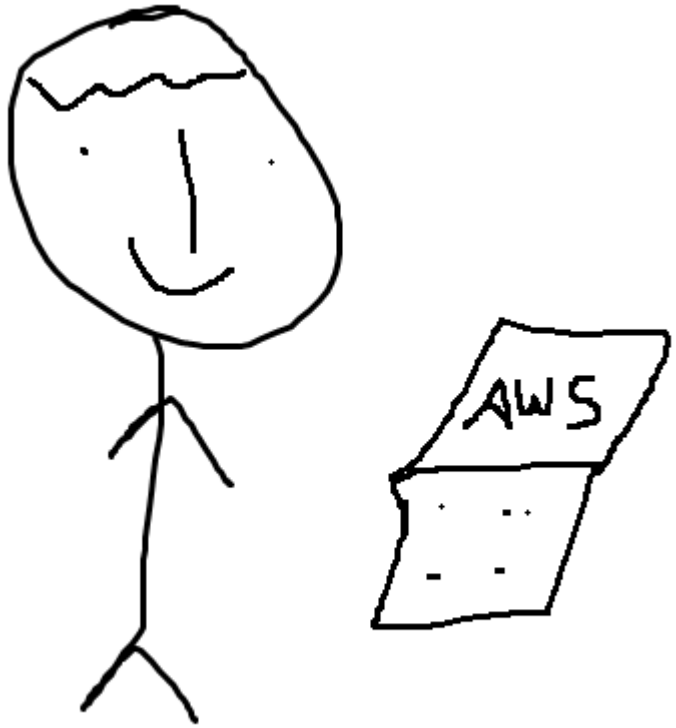
- I look **after Cloud Operations / Security and Backend** at Zebec.
- I build **softwares** and **infrastructure** at Leapfrog Technology Inc.

github.com/regmicmahesh
facebook.com/regmicmahesh

Let’s start building a secure infrastructure on AWS.

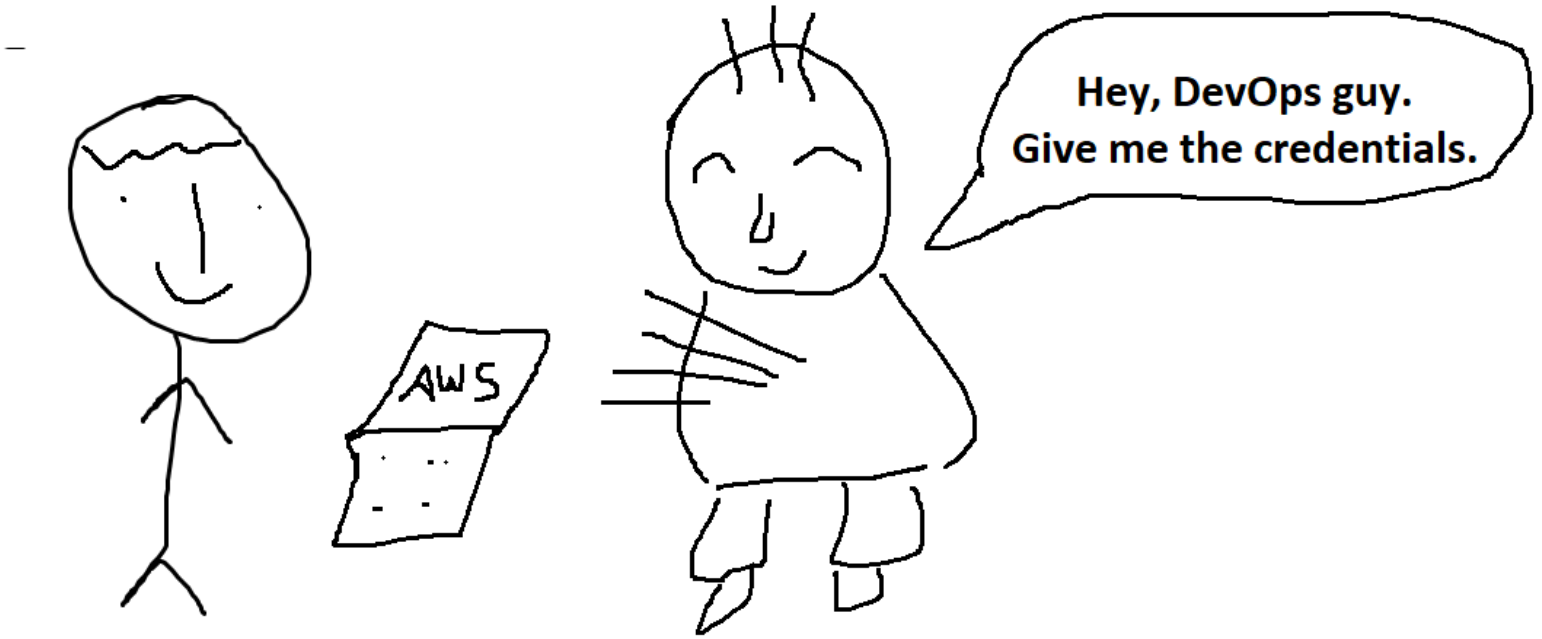


You just created an AWS account for your organization.

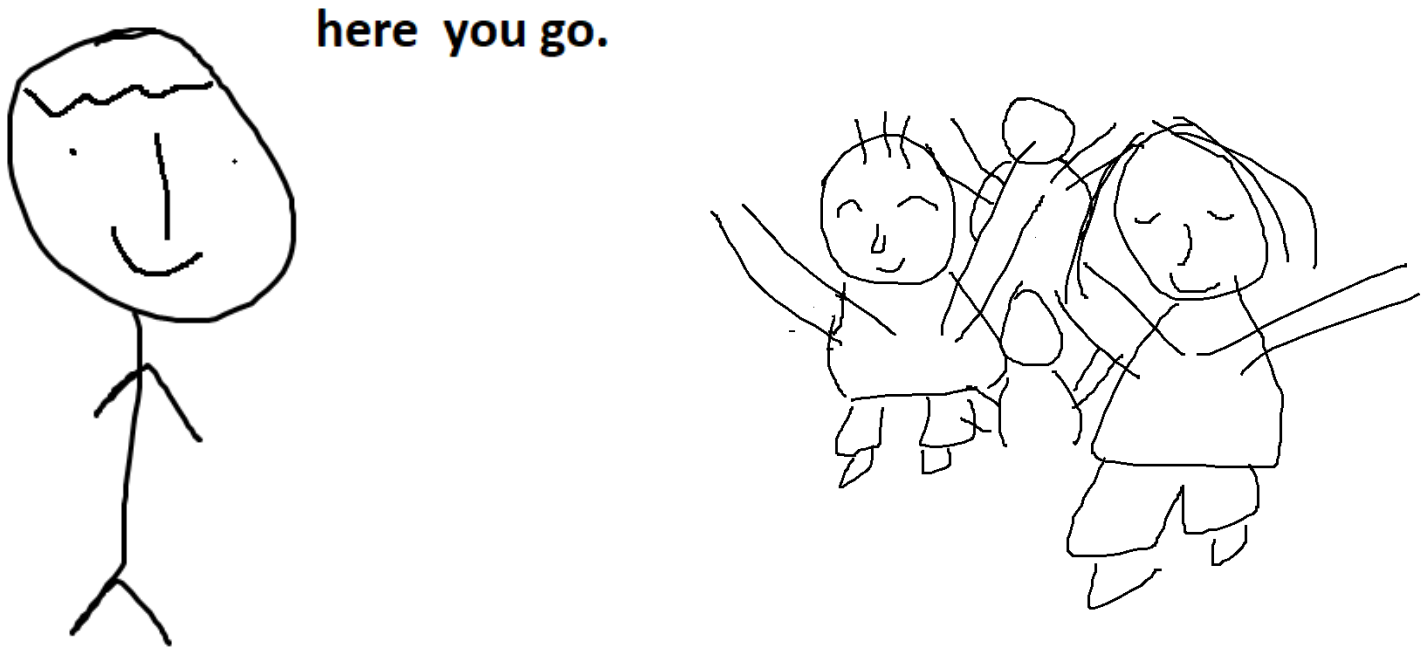


You are the root user of that AWS Account.

Now you need to provide access to another user in your organization.

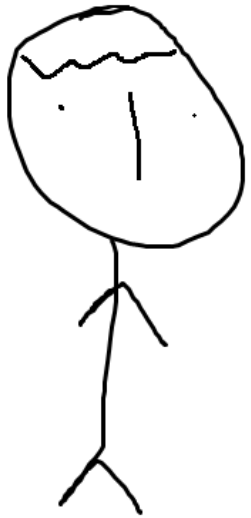


Just like every good devops, you provide your root credentials.



Next day.

hey devops guy, i'm the account manager seems like aws billed us 2000$ last night, what happened?



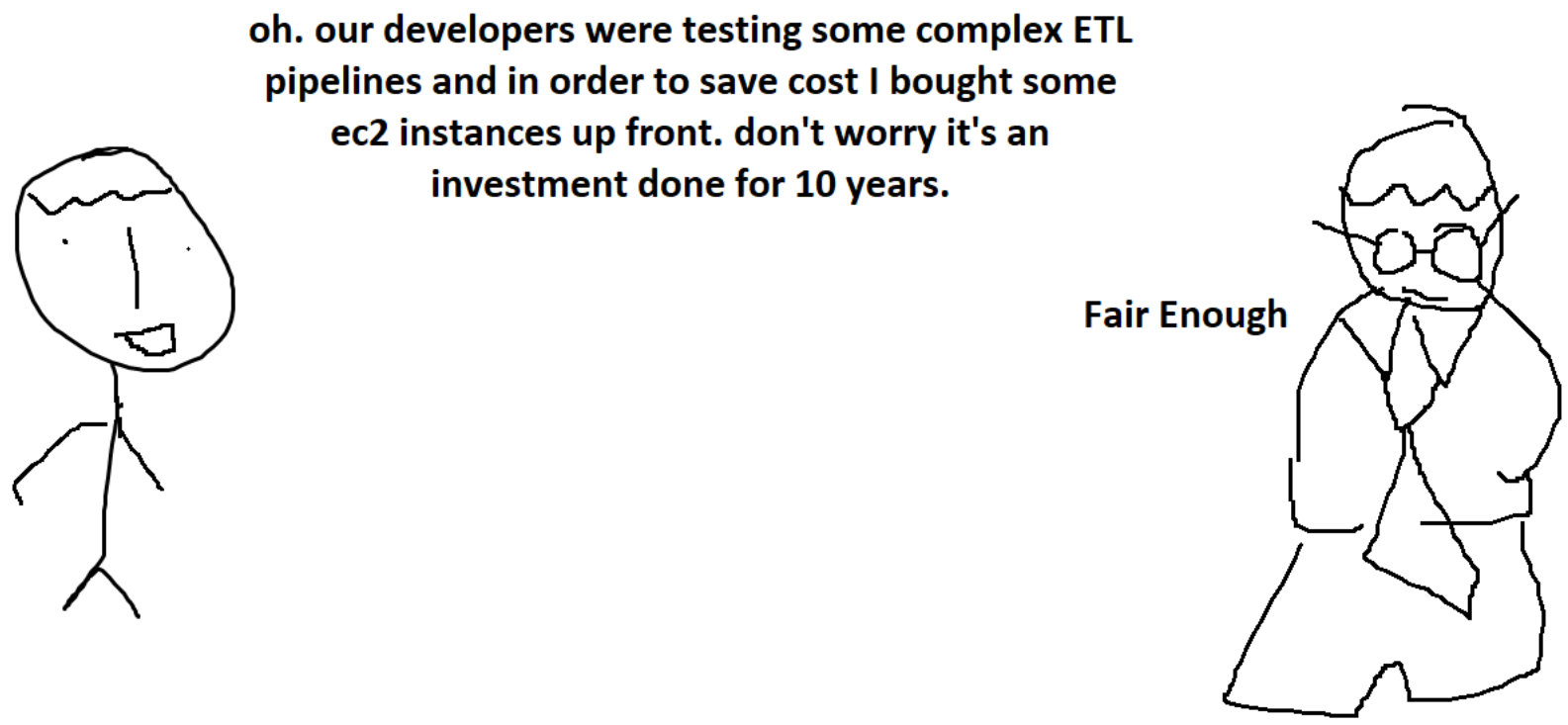
You’re now scratching your head.

I don’t know who created this database with 256GB RAM and 16vCore.

I can’t even remember all the people in my company who have access to AWS.

I can’t even remember all the people in my company who has access to AWS.

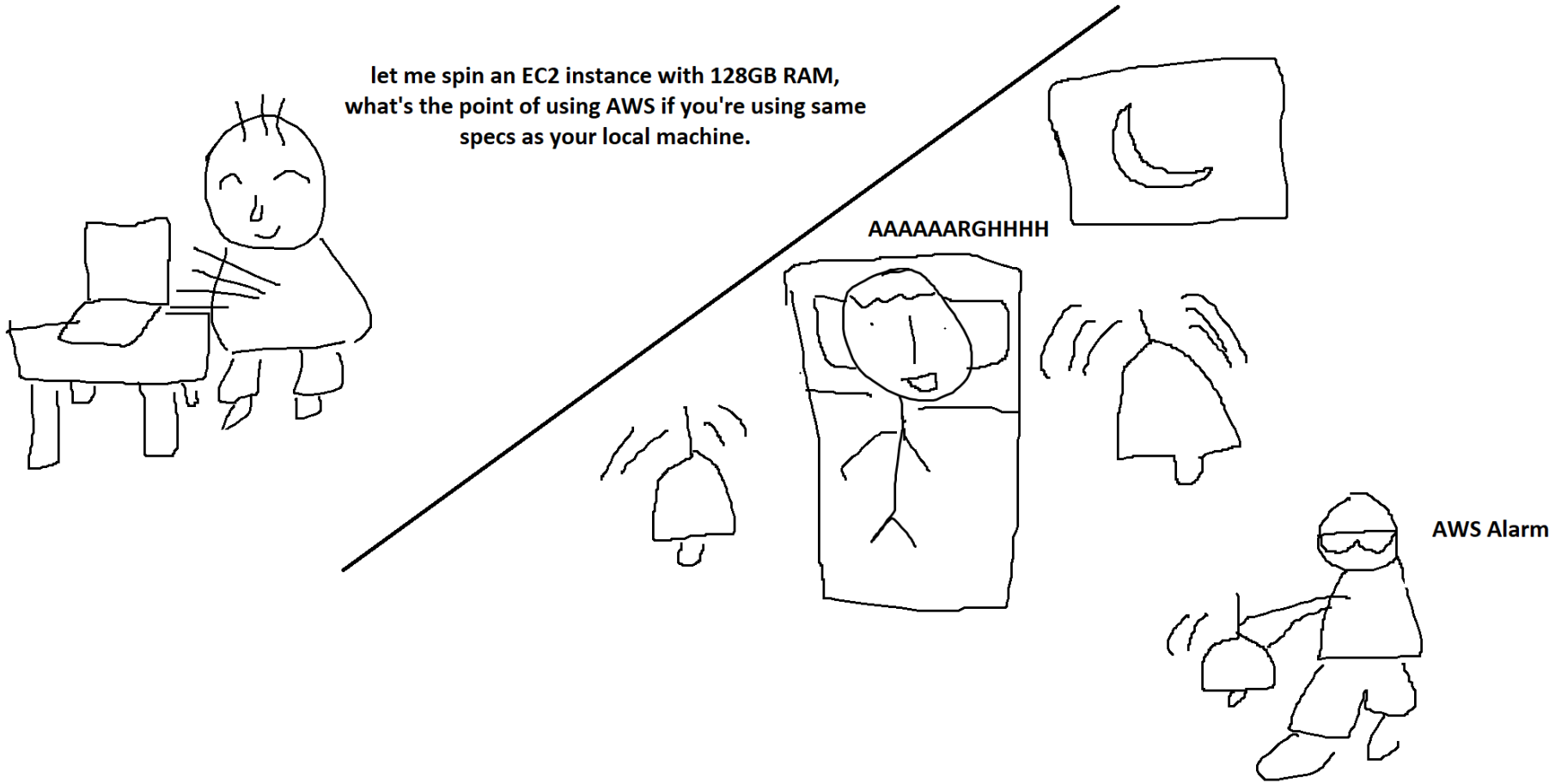
Every DevOps has an ~~excuse~~ failover plans.



Use centralized user repository to manage access properly.

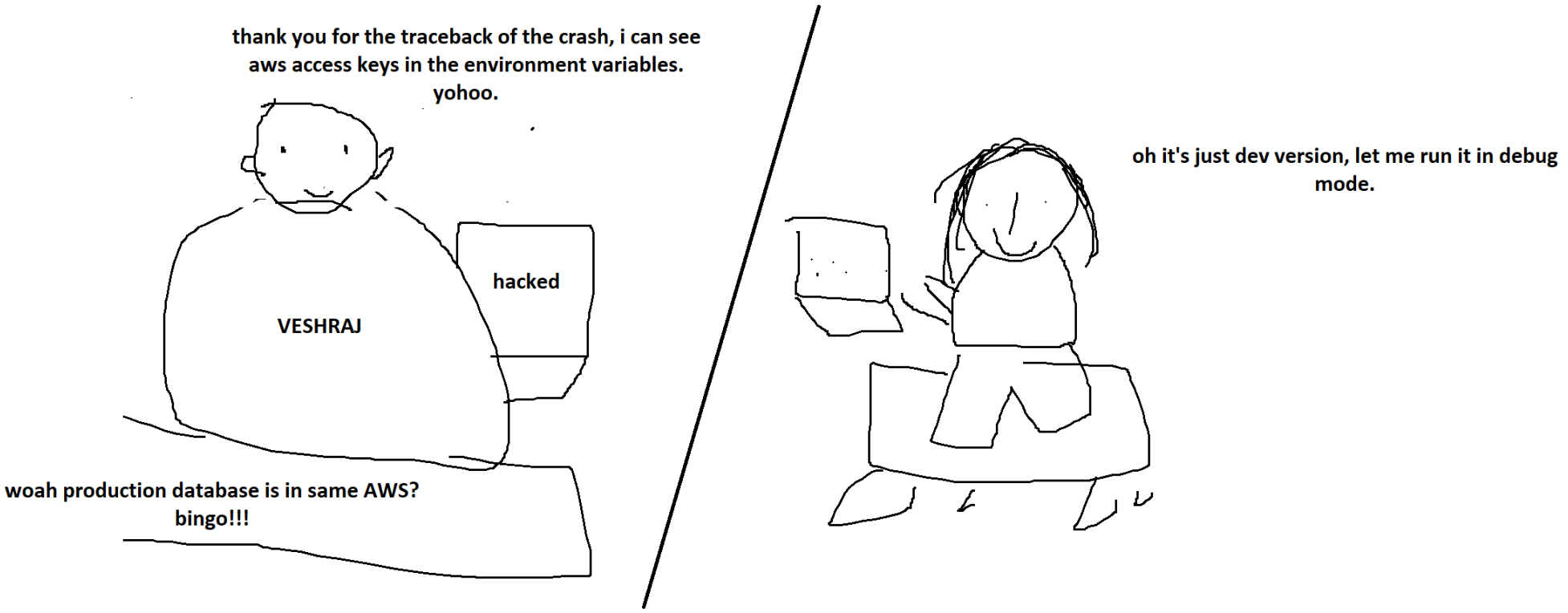
- AWS Directory Services.
- SSO
- Okta
- PingIdentity
- Azure Active Directory

Setup AWS Billing Alarm.



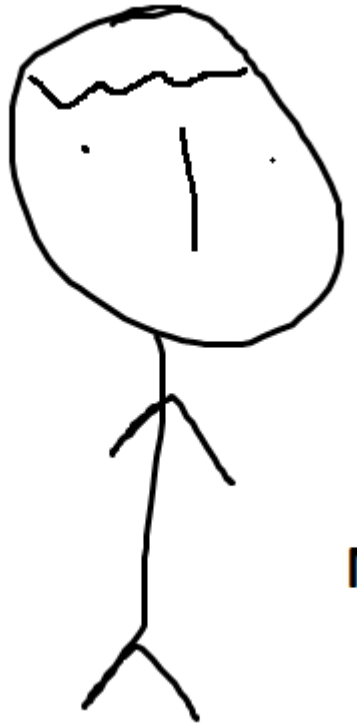
Setup AWS Guard-Duty.

Differentiate dev, staging and prod AWS accounts to reduce blast radius of security breach..



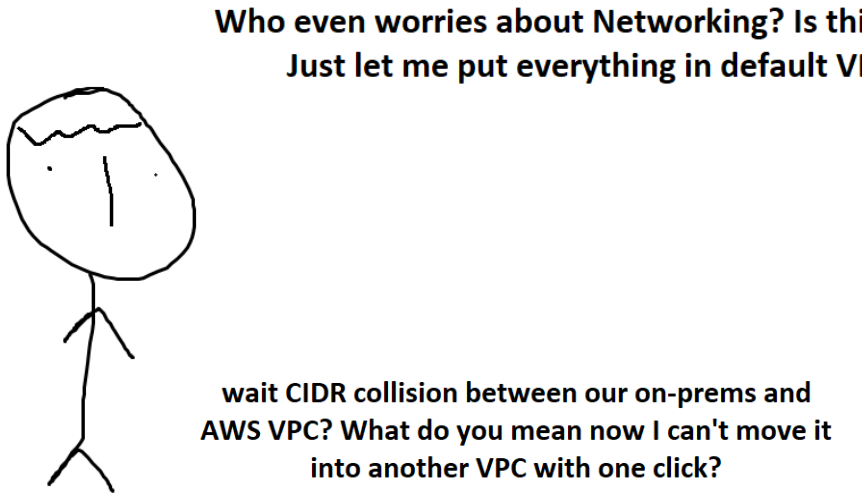
Every developers can be this dumb, I'm no sexist.

Now initial setup is complete.



Now, I can finally deploy some resources.

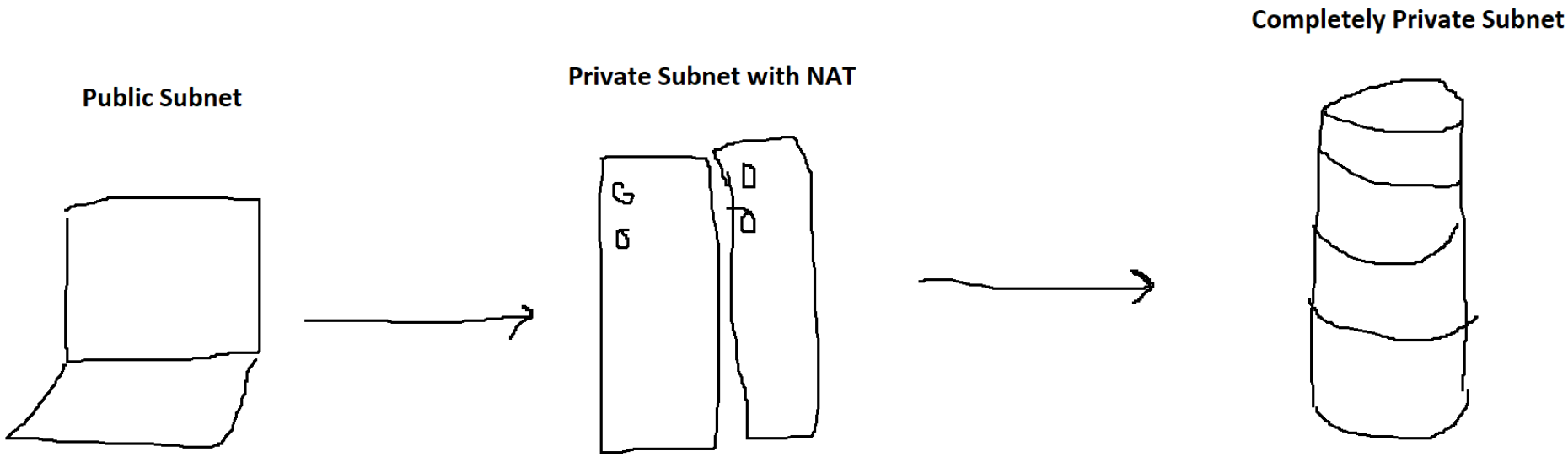
Create your OWN VPC with different CIDR for each environment without collision and **enable VPC Flow Logs**



- No VPC Flow Logs
- Unrestricted NACLs
- No Resource Tagging
- VPC CIDR Collision

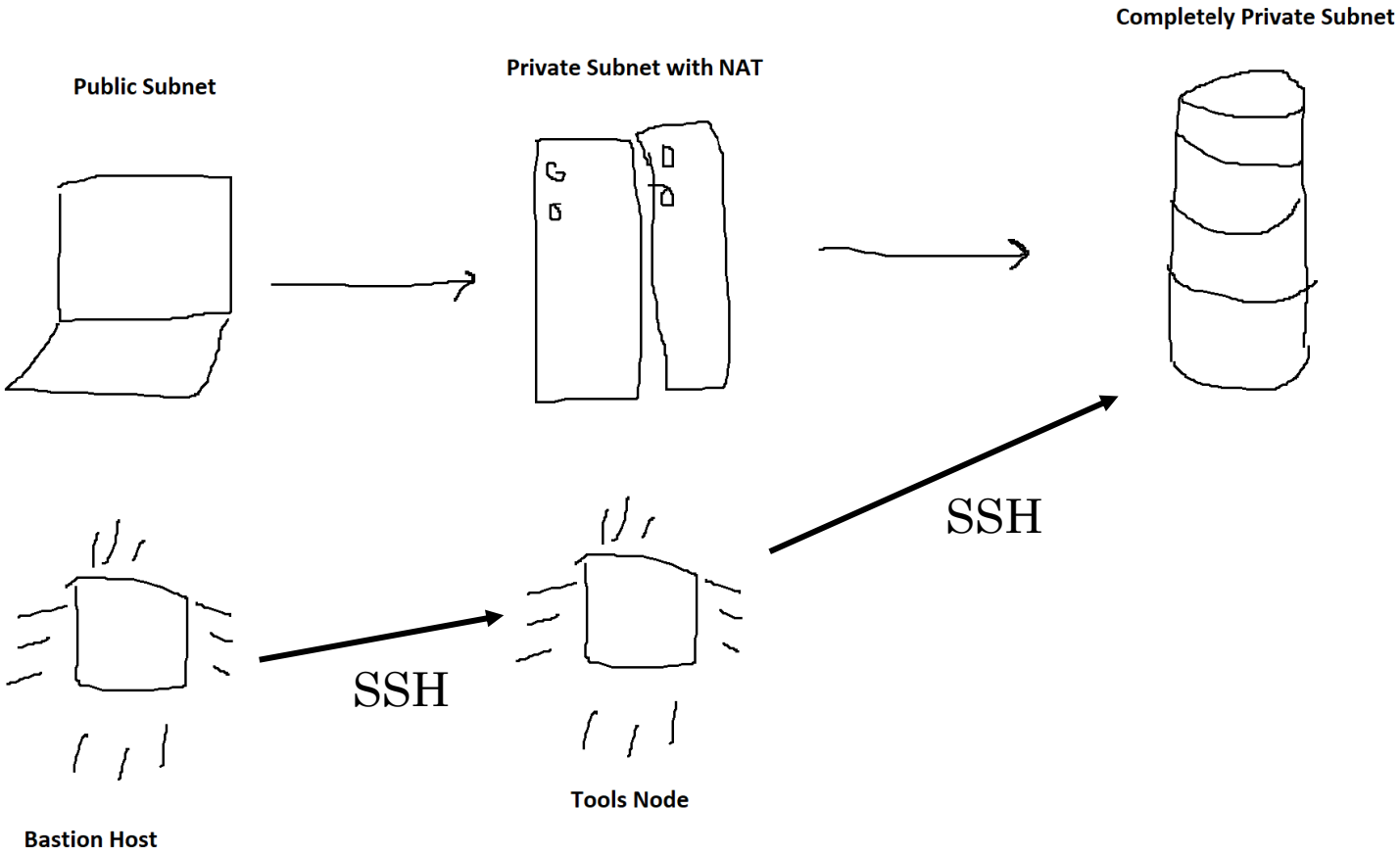
Now deploy everything in public subnet.

Design a 3-tier architecture networking setup.

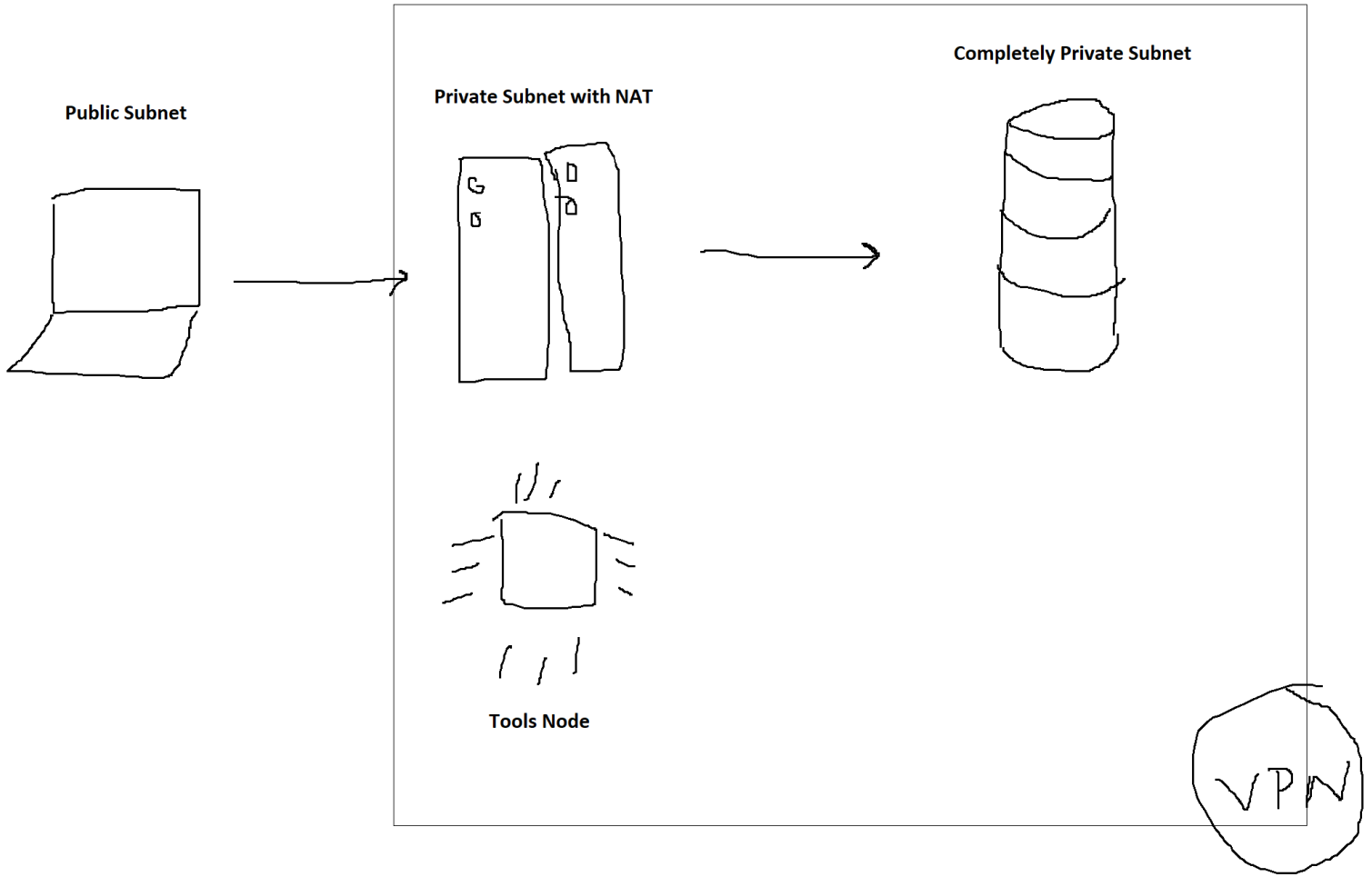


How does one access the database remotely?

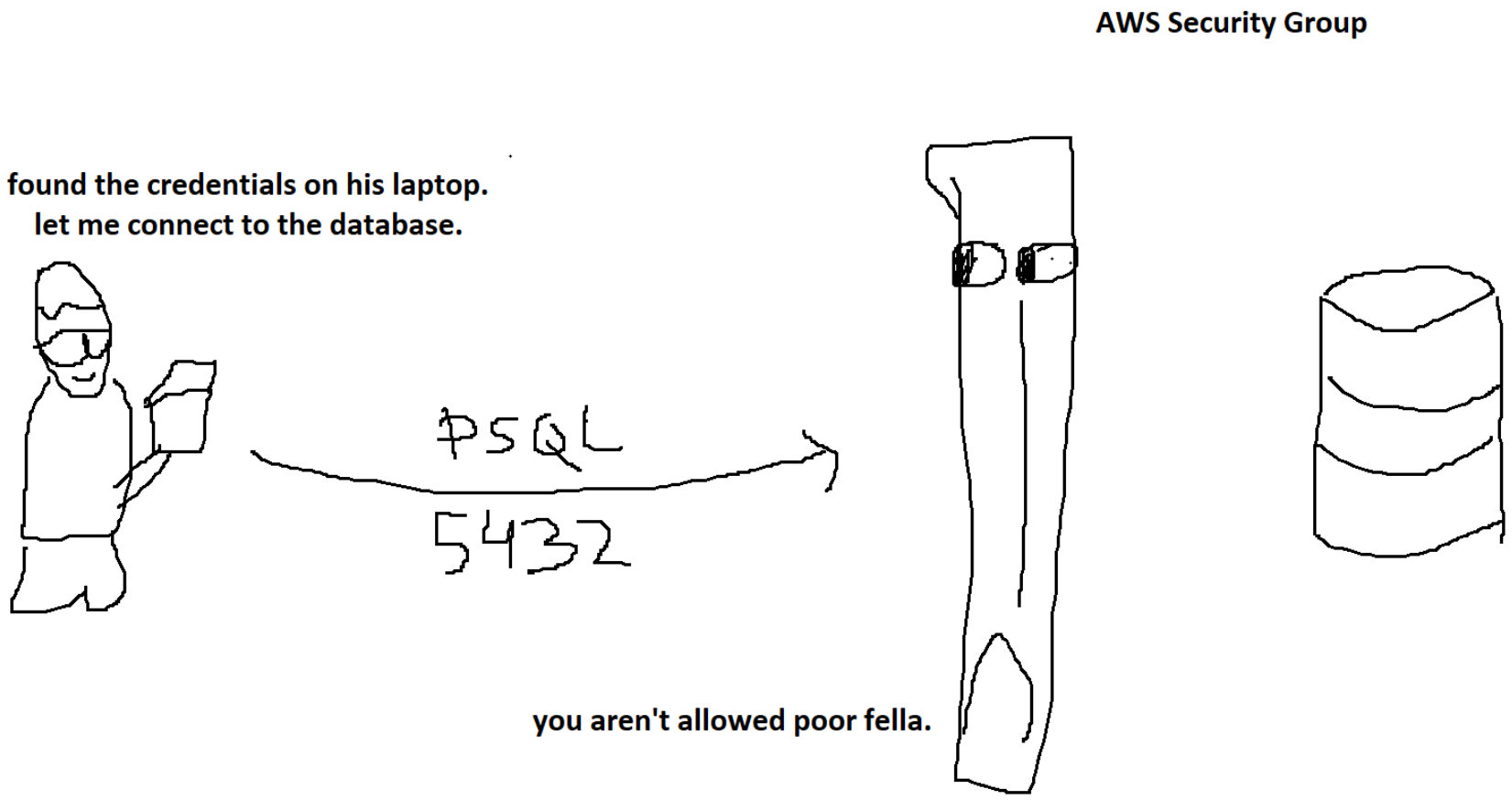
One way.


Completely Private Subnet
Public Subnet
Private Subnet with NAT
SSH
SSH
Tools Node
Bastion Host

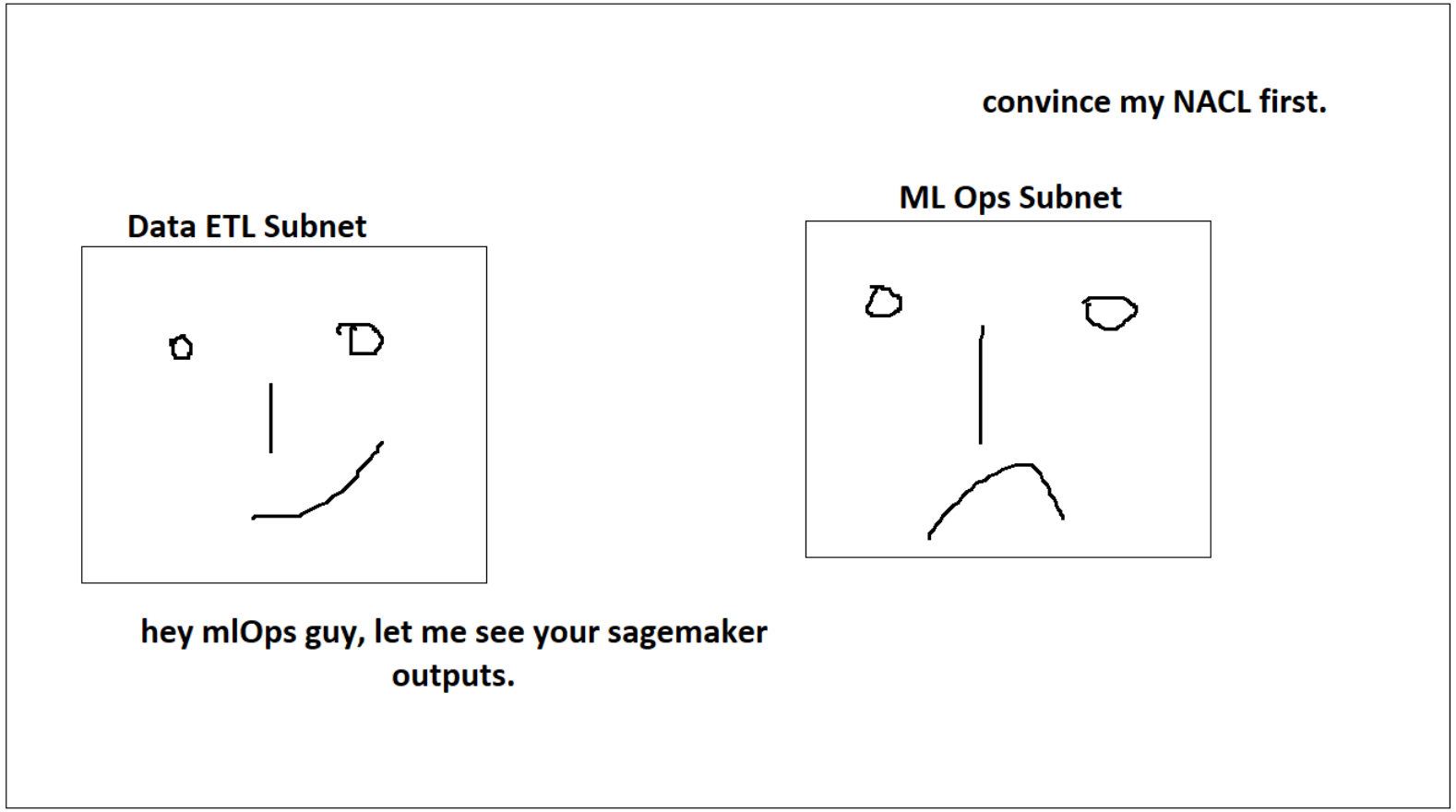
Better way using AWS Client VPN.

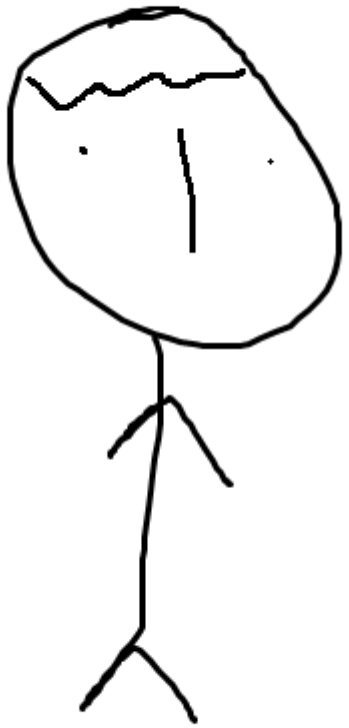


Don’t be over-permissive on ports.



Use NACL to filter traffic across subnets.

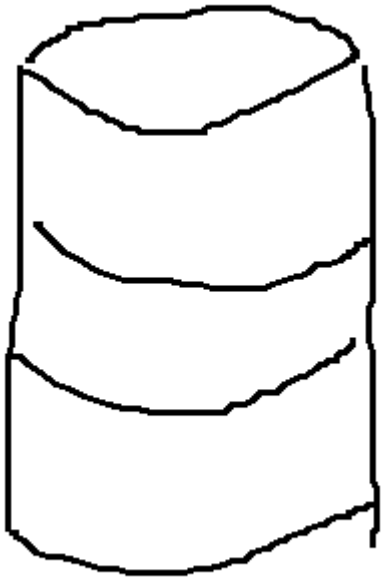




Network is ready.

Time to deploy resources now.

First comes database.



Put it in a private subnet without NAT.

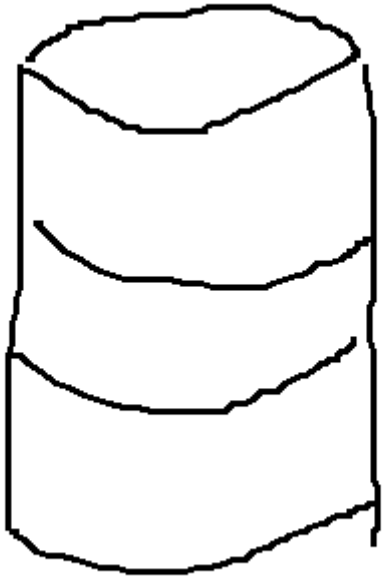
Enable Automated Backups.

Use IAM Based Authentication as much as possible.

Don’t share default database credentials.
Save it in AWS Secrets manager with explicit deny.

Encrypting Data in RDS

Automatically uses KMS
(Encryption at rest)



Disable plain TCP connections.
Allow only TLS.
(Encryption in transit)

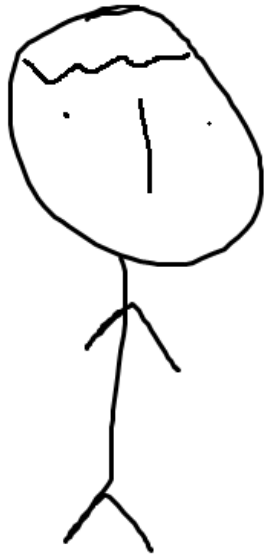
Storing in S3

Store data using Server Side Encryption at-least.

Don’t store sensitive data in S3.
Fine-grained access control is tough in S3.

Enable versioning or object lock.

S3 misconfiguration is very common.

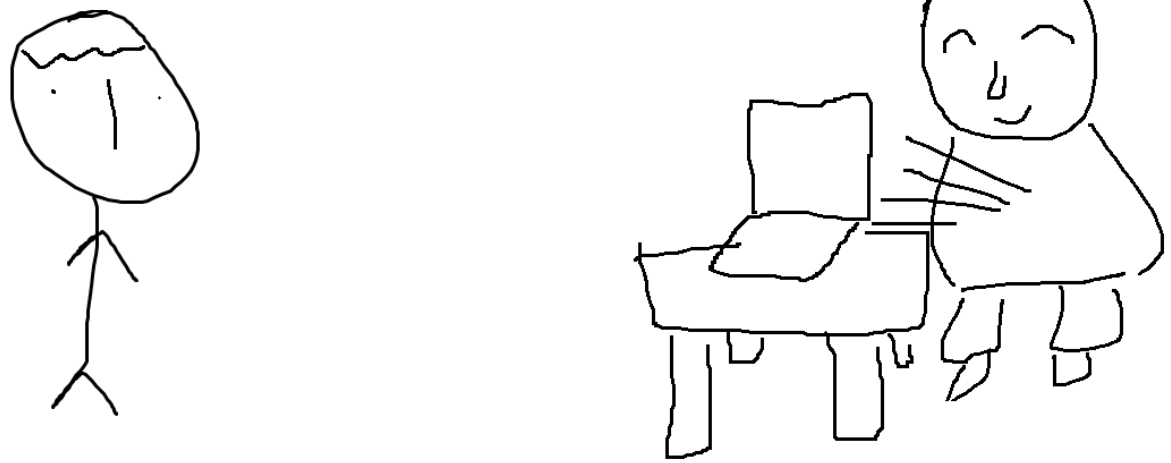


Say with me
"I will never enable ACLs in S3"

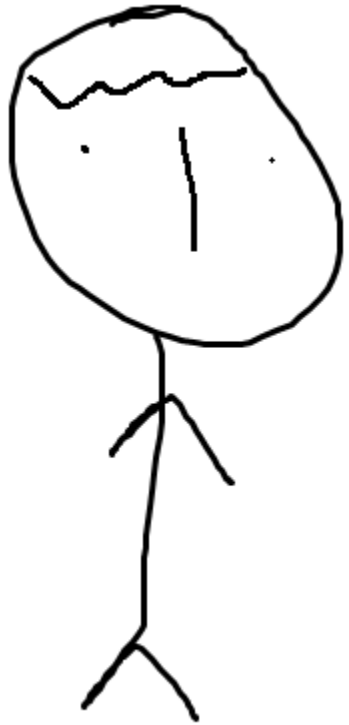
"i won't make anything public in s3 directly"

You just made 5% bug bounty hunters jobless.

Then, how do you host public assets?



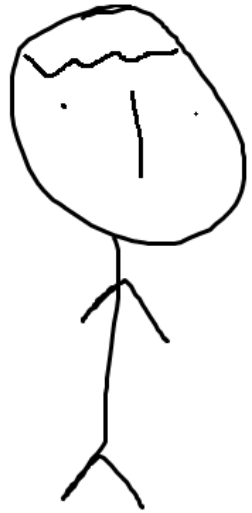
Use Cloudfront.



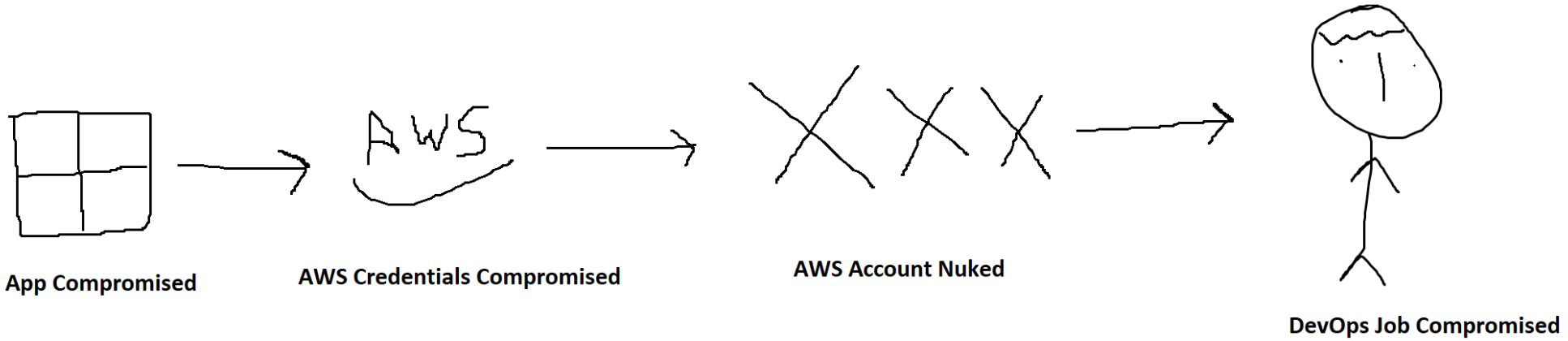
Time to deploy applications.

Deployment with AWS ECS

- Task Execution IAM Role and Task IAM Roles are not the same thing.
- Provide access to AWS Services for your application using Task IAM Roles.
- Task Execution IAM Roles is just for **pulling image** and **pushing logs**.

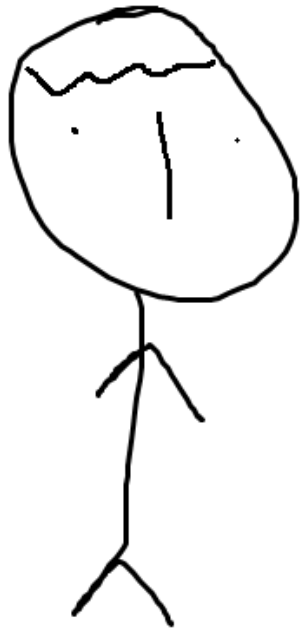


Don’t ever use access key inside of application.



Why IAM Roles for applications ?

- Nobody can assume it except the associated service.
- The tokens are automatically handled and short-lived.
- You don’t need to manage access keys.

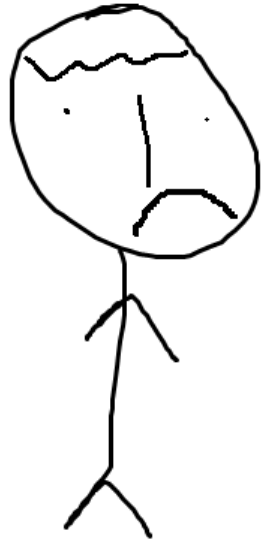


There are two major things to secure.

- Environment Variables and Secrets.
- Container Images.

First Way.

- Pass Secrets as Environment Variables in ECS Task Definition.

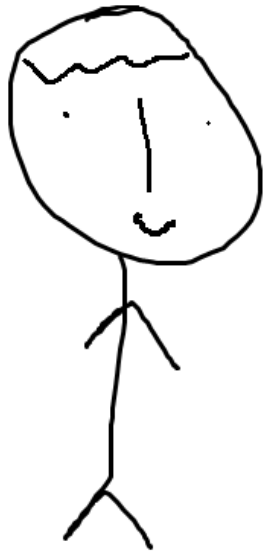


Better Way.

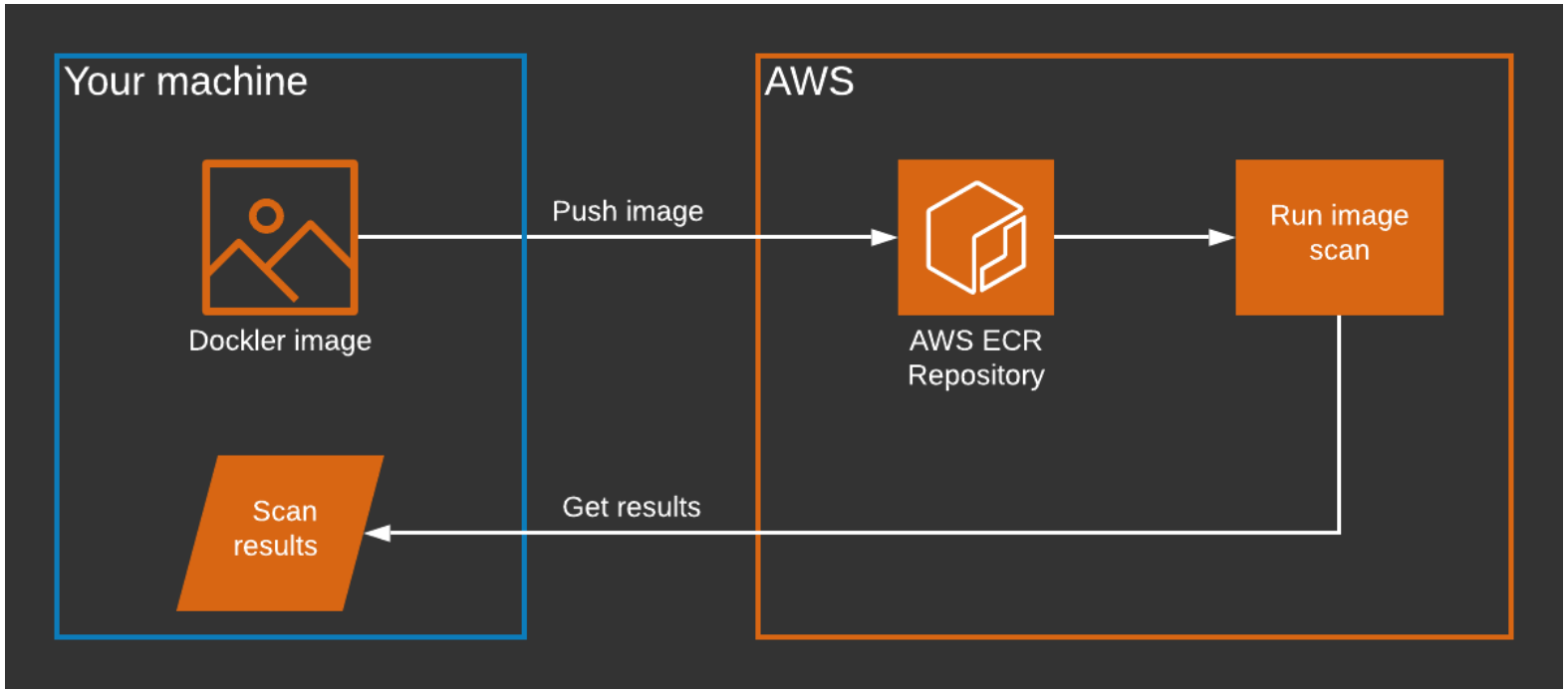
- Use bootstrapping scripts to populate the keys before running application.

Best Way.

- Use ‘valueFrom’ to pull the appropriate keys from AWS Secrets Manager or Parameter Store.



Always enable scan image on push on ECR.



Some container security tips.

- Don’t run container as root user.
- Always add a readiness check and healthcheck probing mechanism.
- Do not store secrets in dockerfiles.
- Try to keep the image as small as possible.
- Use trusted base images.

Securing your application in transit.

SQL Injection Payloads

File Inclusion Payloads

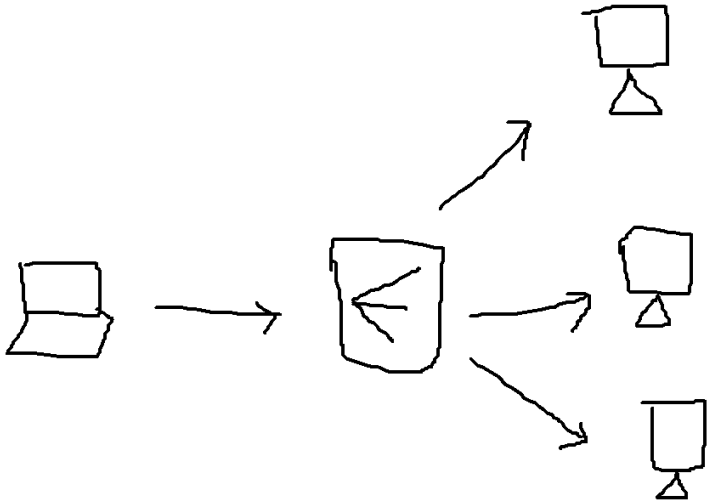
XSS Payloads

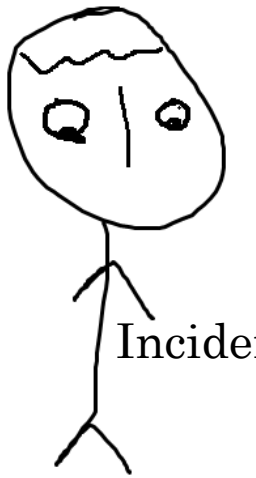
Fuzzers

Suspicious File Uploads

Host Header Injection Attacks

Always have load balancer when facing internet.



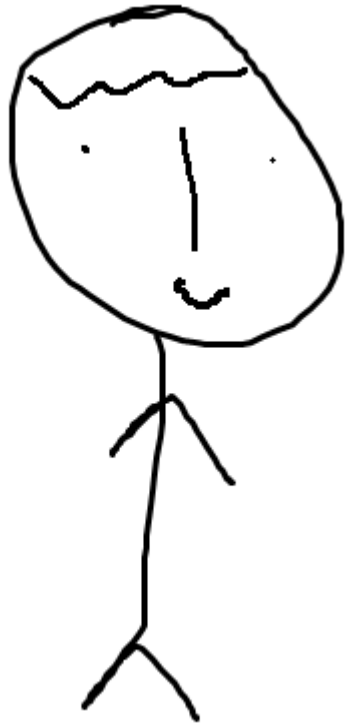


Elastic Load Balancer can block invalid HTTP(S) requests from reaching to the backend

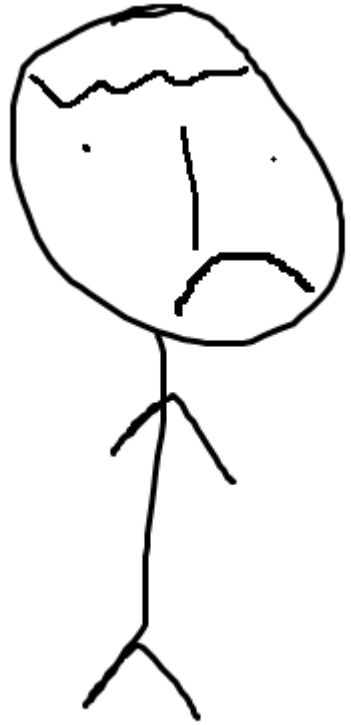
Incident: Some fuzzers generate very long path and the web framework threw an exception causing the backend process to crash.

AWS Shield Standard

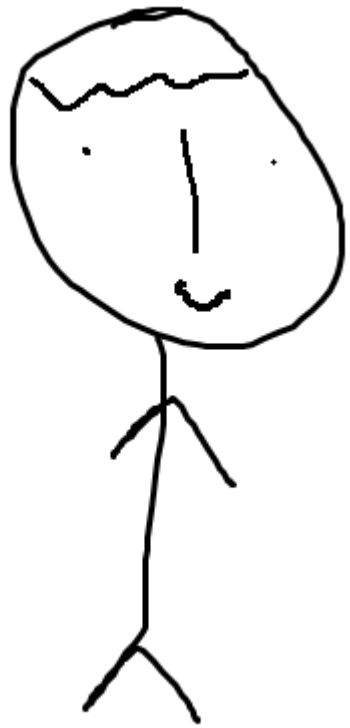
Now our infrastructure is safe and secure.



But what about the application?



I’m a DevOps I don’t get paid enough to be worried about that.



Checks in Deployment Process

- Pre-commit hooks can can different types of errors / logical flaws even before commiting a change to the git repository.
- Integrate tools like Synk / Sonarqube to scan the repo in the CI process to detect if there is any potentially vulnerable codes.
- Be involved in the **development** process.

Now let's get into a interesting part which I've implemented recently.

Access Key Honeypots.

1. Create IAM User with no privileges.

2. Drop it into critical systems as environment variables.

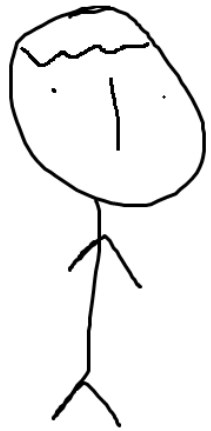
3. Continuously monitor cloudtrail to see if there's any activity from that user.

4. If there is, your system got breached.

How to trigger this?

- Send this access key to everyone in your organization and request to add it in production systems. Don't mention any project.

- If they're breached, the hacker will try to do something out of it.



Securing complete infrastructure takes much more than this but this should at least point you in the right direction.

Thank you!