**AWS Zero Trust Architecture Setup**

**📌 Project Overview**

This project implements a **Zero Trust security model on AWS**, designed using a Bastion Host and an App Server in separate subnets. The core idea behind Zero Trust is “Never trust, always verify” — and this setup enforces strict access control at every level of the infrastructure.

**🧱 Architecture Components**

* **Bastion Host**: Located in a public subnet with a public IP. Acts as a secure entry point.
* **App Server**: Placed in a private subnet with no public access. Accessed via the Bastion Host only.
* **VPC and Subnets**: Custom VPC with public/private subnets in the same availability zone.
* **Security Groups**:
  + Bastion Host SG allows SSH from a trusted IP.
  + App Server SG allows SSH only from Bastion Host's private IP range.
* **IAM Policies and Roles**: Minimal privileges with clearly scoped permissions.
* **SSH Key Pair**: Secure login using .pem file and no password access.

**🔐 Key Features**

* Enforces **least privilege principle**
* No direct internet access to the app server
* IAM identity control instead of relying on traditional per-device trust
* Secure resource segregation via subnets and SGs
* SSH tunneling using Bastion (jump host) strategy

**⚙️ Technologies Used**

* AWS EC2 (Amazon Linux 2023)
* AWS VPC & Subnetting
* IAM Roles & Policies
* SSH and .pem key-based authentication

**🚀 Steps Performed**

1. Created VPC with CIDR block 10.0.0.0/16
2. Created public and private subnets (10.0.1.0/24 and 10.0.2.0/24)
3. Launched Bastion Host in public subnet (with internet gateway)
4. Launched App Server in private subnet (no public IP)
5. Created security groups:
   * SSH to Bastion from my IP only
   * SSH to App Server from Bastion’s private IP only
6. Connected to Bastion via .pem key → then used SSH to reach App Server
7. Applied IAM roles to restrict and manage access securely

**💡 Learning Outcome**

This project helped me understand Zero Trust implementation on AWS, the importance of IAM security, subnet-level isolation, and how to architect secure infrastructure using AWS-native tools.

**🧠 Future Improvements**

* Integrate AWS Systems Manager Session Manager (for even more secure access)
* Add logging and monitoring with AWS CloudWatch
* Automate with Terraform or CloudFormation