Demo Project: Configure a Shared Remote State

This guide outlines how to configure Amazon S3 as remote storage for Terraform state. Using a shared remote state ensures all team members have access to the latest state data and eliminates the need to store state locally or within the Jenkins server.

Step 1: Modify `main.tf` to Use S3 Backend

1. Open your Terraform configuration file `main.tf`.
2. Add the following backend configuration to specify the remote S3 state:

   Note: the bucket name needs to be unique.

```
terraform {
  required_version = ">= 0.12"
  backend "s3" {
    bucket = "myapp-tf-eb-s3-bucket"
    key = "myapp/state.tfstate"
    region = "us-east-1"
  }
}
```

Step 2: Create the S3 Bucket

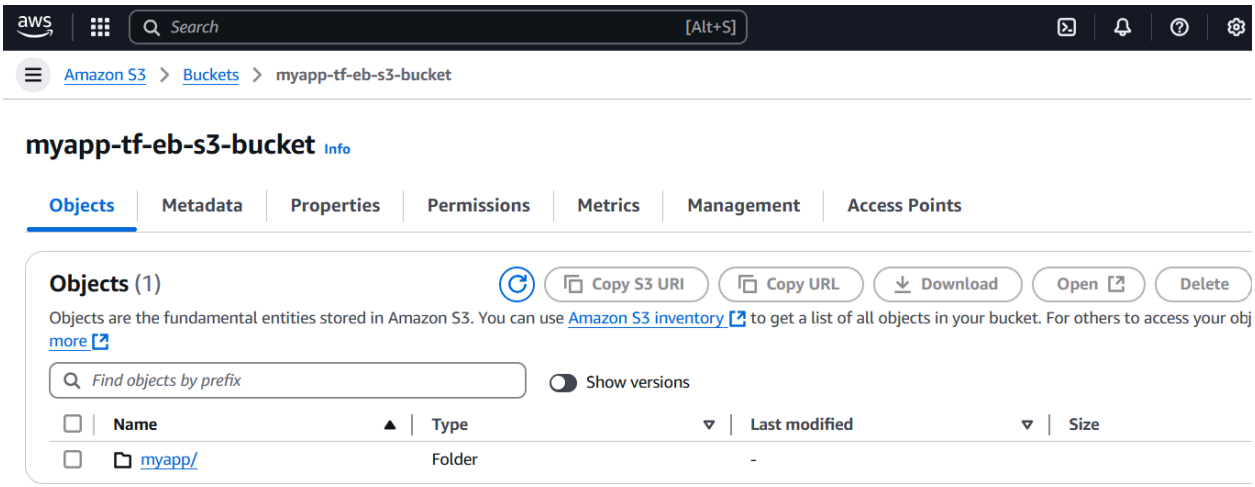
1. Log in to the AWS Management Console.
2. Navigate to the S3 service.
3. Create a new bucket:
    - **Bucket Name:** `myapp-tf-eb-s3-bucket` (or another unique name)
    - **Region:** `us-east-1`
4. Enable bucket versioning (recommended for recovery purposes):
    - Select the bucket.
    - Go to **Properties**.
    - Enable **Bucket Versioning**.

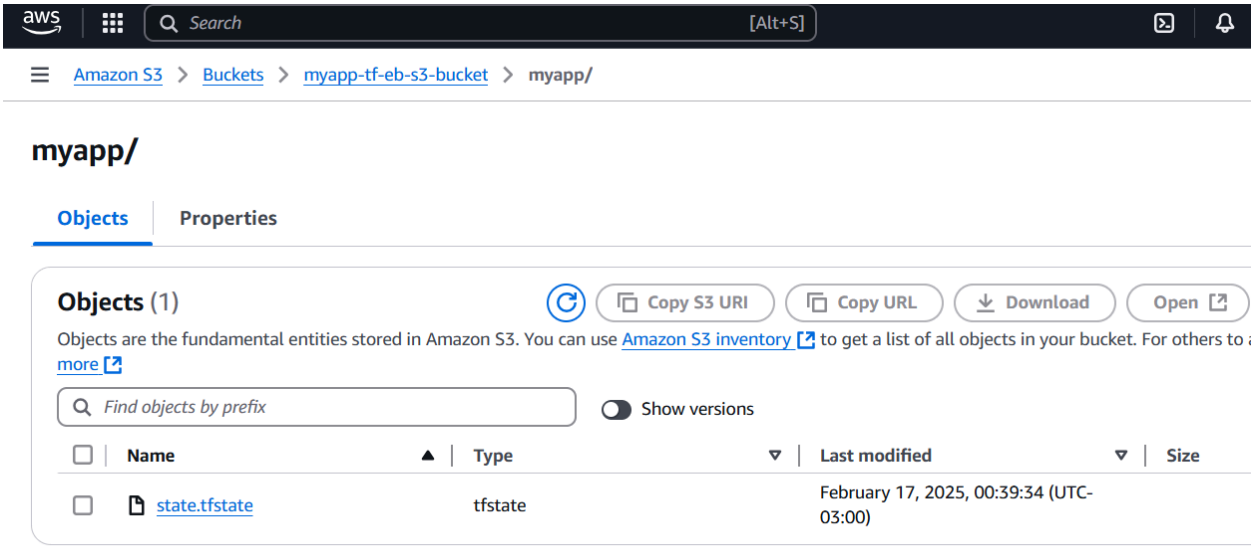
Step 3: Commit and Push Changes

1. Save the changes to your `main.tf` file.
2. Commit and push the changes to your Git repository.
2. Trigger your CI/CD pipeline.

Step 4: Verify the State File in S3

1. Once the pipeline completes, navigate to the S3 bucket in the AWS console.
2. Verify that the `state.tfstate` file is present under the specified key (`myapp/state.tfstate`).





Step 5: Access Shared Remote State

With valid AWS credentials, you can access the shared remote state from any machine.

1. Navigate to the Terraform directory: `cd terraform`
2. Initialize Terraform: `terraform init`
3. List all resources in the state: `terraform state list`

This command will connect to the S3 bucket and display a list of all resources managed by Terraform.

Step 6: Clean Up Resources

1. Run the following command: `terraform destroy --auto-approve`
2. Ensure all resources are destroyed.
