Terraform: Moving Existing Resources to Subfolders & Managing State

This document provides guidance on how to safely move existing Terraform resources into subfolders (modules) and manage the state effectively. It includes two approaches—using Terraform's `moved` blocks (recommended) and manual `terraform state mv` commands.

Option A (Recommended): Using `moved` Blocks (Terraform ≥ 1.1)

----

1. Create a subfolder module structure and move the resource configuration.
2. Add a module block in the root.
3. Add a `moved` block in the root to remap state addresses.
4. Run: terraform init → plan → apply.

Example moved block:

```
moved {
  from = aws_s3_bucket.logs
  to   = module.storage.aws_s3_bucket.logs
}
```

Option B: Using `terraform state mv` (Manual)

----

1. Move the code into the module.
2. Add the module block in root.
3. Move the state entry manually:

```
terraform state mv aws_s3_bucket.logs module.storage.aws_s3_bucket.logs
```

4. Validate with terraform plan.

Managing the State File (Backend)

Case 1: Same backend — no special steps required.
Case 2: Splitting monolith into multiple stacks — create new roots, initialize backends,
        and use import/state mv operations.
Case 3: Changing backend — update backend configuration and run:

```
terraform init -migrate-state
```

Best Practices & Tips

• Always back up the state: terraform state pull > backup.json
• Use `moved` blocks for long-term maintainability.
• Plan after every step to avoid unexpected resource recreation.
• When splitting stacks, migrate/import resources carefully.
• Use DynamoDB locks with S3 backends.

End of Document
