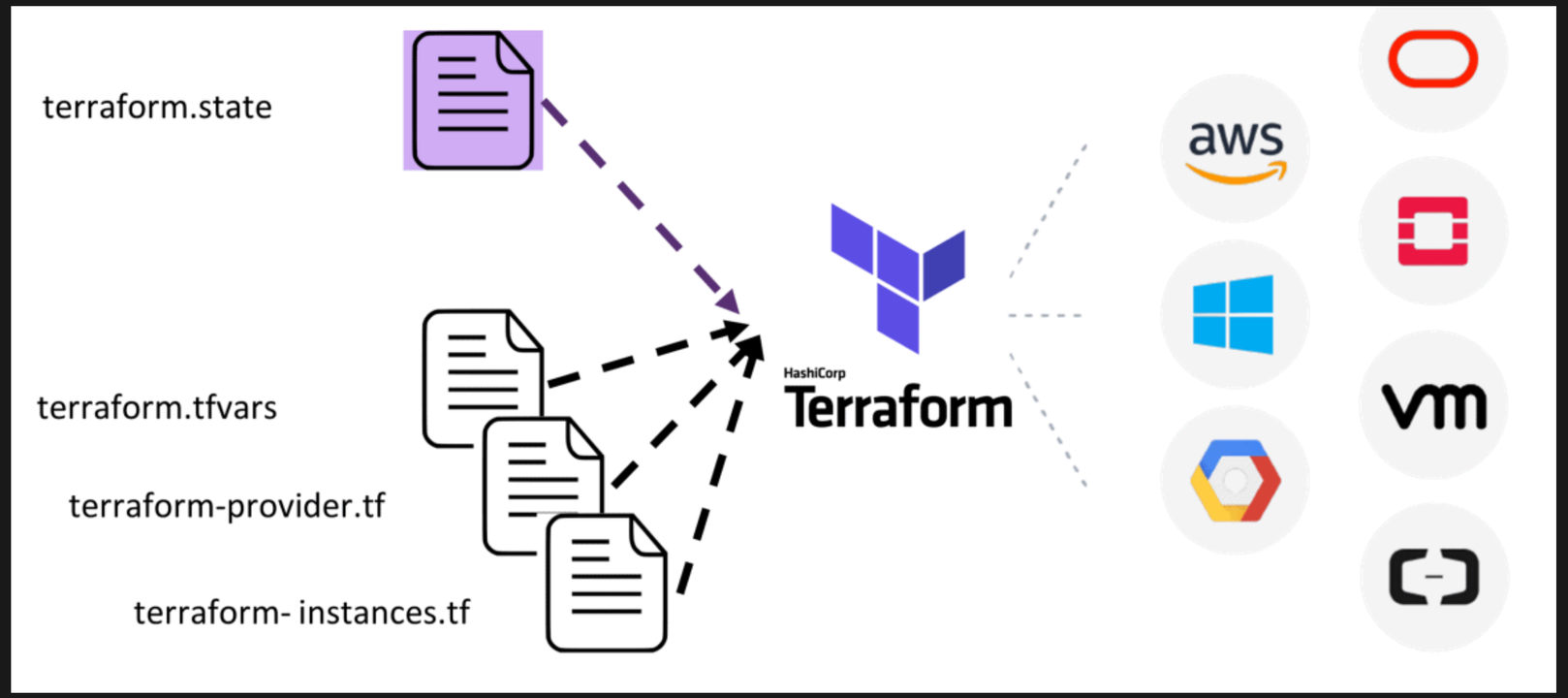
Terraform State Remote Backend

HashiCorp Terraform

State Backend

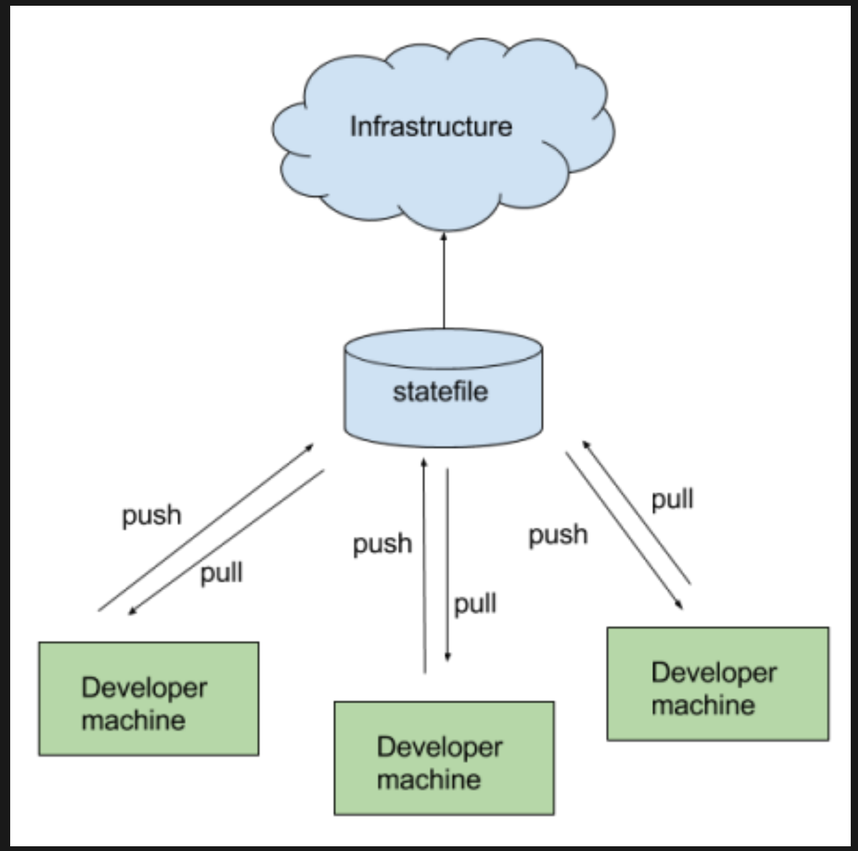
- There are two types of States. Local and Remote
  - The types refer to how the states are being stored
- Local states are stored in the local directory of your project but poses many problems
- You always want to use remote states whenever possible
- How does a state become remote?
  - States are defined as being remote when the statefile is stored in a centralized storage, outside of your repository
  - S3 is a common object storage solution, perfect for storing your statefiles

State Backend



State Backend

- Remote states are stored elsewhere and is persisted
- Resolves the painpoints of using local state
  - Easily allows others to work on Terraform - highly scalable as state is managed elsewhere
  - State locking mechanisms protect from race conditions
  - Sensitive data is no longer exposed in repository
- Terraform provides State locking out of the box
  - Similar to transactions in SQL or file locking in Linux
  - State locking ensures that only one entity is writing into the state at any time
  - Must enable state locking - depends on which storage solution used
  - AWS uses DynamoDB



HashiCorp Terraform

State Backend

- When remote states are stored in a central location outside of your repo
  - We call this, a State Backend
- S3 is a very common solution for storing state, provides advanced features, ideal storage solution
  - Enabling versioning protects statefile from accidental deletion
  - Easily rollback and provide strict permissions
- Next lecture - configuring a state backend to use S3