Ansible Flow

Our ansible setup is a bit difficult to follow because:

- it starts / stops
- defers functionality to other tools / platforms (terraform & circlieci)
- passes variables through to those other tools / platforms
- updates its own runtime based on the outputs of these tools / platforms
- runs the same code modules at different points, using flags to trigger different branches of execution

So I've tried to document it in detail. This markdown below charts the flow of execution from a file perspective. But misses the nuance of how some modules embed others

Files:

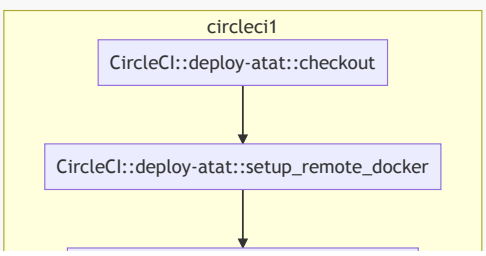
- site.yml
    - infra.yml
        - role: crypto
            - Create the dh directory if it does not exist
            - Generate Diffie-Hellman parameters with the default size (4096 bits)
        - role: bootstrap
            - fetch atat.pem
            - fetch bootstrap vars
            - **if** fetch_bootstrap_state # default false
                - fetch bootstrap state
            - get vars
            - **if** deploy_bootstrap_env # default true
                - bootstrap env
            - **if** show_bs_env # default true
                - show outputs from run
            - **if** deploy_bootstrap_env # default true
                - push local state to remote
        - role: circleci
            - **PARAMS**
                - build_and_push_ops_image - True
            - get the circle ci api token
            - get tf outputs
                - **PARAMS**
                    - deploy_app_env: false
                - **WTF** - include_role - terraform - what does this do? - Does this now load terraform's role, and just pass the invluded variable?! Guess that means the defaults here also apply and could override terraform? Shoot me.
            - **if** build_and_push_ops_image
                - run circle ci ops build
                    - Kick off a build for image
                        - ** ============= RUNS ON CIRCLECI ============= **

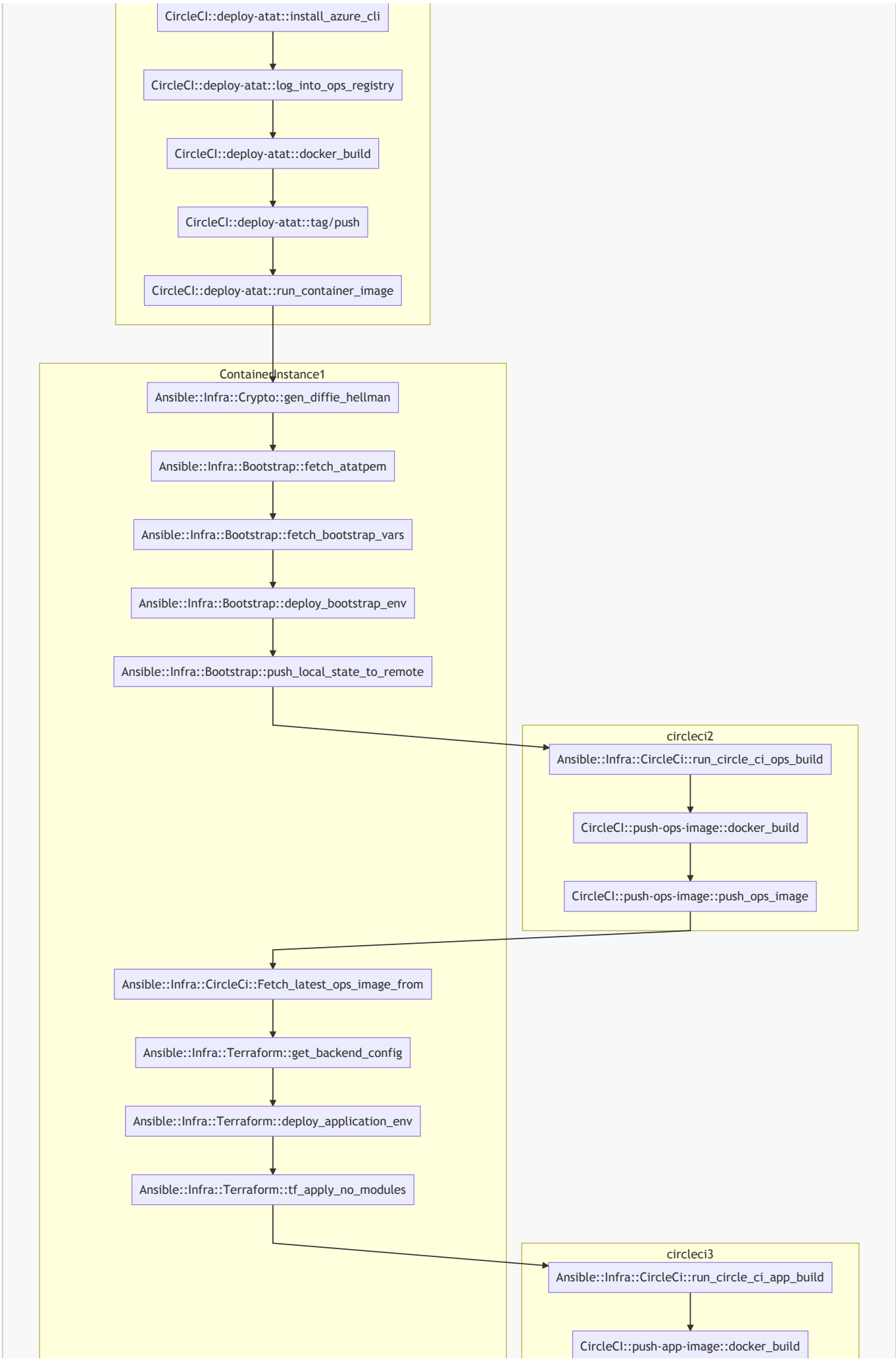
        - Fetch latest `ops` image from
            - Polls, waiting on circleci to complete
    - Note, also has build_and_push_app_image, but this is disabled
- role: terraform
    - get backend config
        - **TODO: How Does This Work**
            - with_items: "{{ tf_bootstrap_backend_outputs }}"
            - when:
                - item != 'environment'
                - item != 'ops_container_registry_name' -->
        - build backend
            - SETS the b_end fact, such that it will presumably do the real terraform call later
    - get backend config (dupe name)
        - Download tfvars file
            - **TODO** Security check - is this saved to an encrypted drive?
                - /src/terraform/providers/application_env/app.tfvars.json
    - get variables
        - **TODO:** This is all just variable setting - don't really follow the commands or intent yet
        - set tf vars file path
        - pull in tfvars values
        - make k/v
    - **if** deploy_app_env
        - deploy application env
            - set tf_dir # Set fact
            - **if** target_modules is defined **AND** b_end is defined
                - tf apply (modules)
                    - **Note:** modules means only install the selected modules. It does LESS than no modules, not more
            - **if** target_modules is **NOT** defined **AND** b_end is defined
                - tf apply (no modules)
    - **if** show_app_env_outputs
        - show outputs
- role: circle_ci
    - **PARAMS**
        - build_and_push_app_images = True
    - **if** build_and_push_app_images
        - run circle ci app build
            -
                - Kick off a build for image
                - ** ============= RUNS ON CIRCLECI ============= **
    - See above for the other circle ci tasks
- **if** teardown_bootstrap_env
    - role: destroy_bootstrap
        - get vars
            - set those same weird variables there is a todo for above

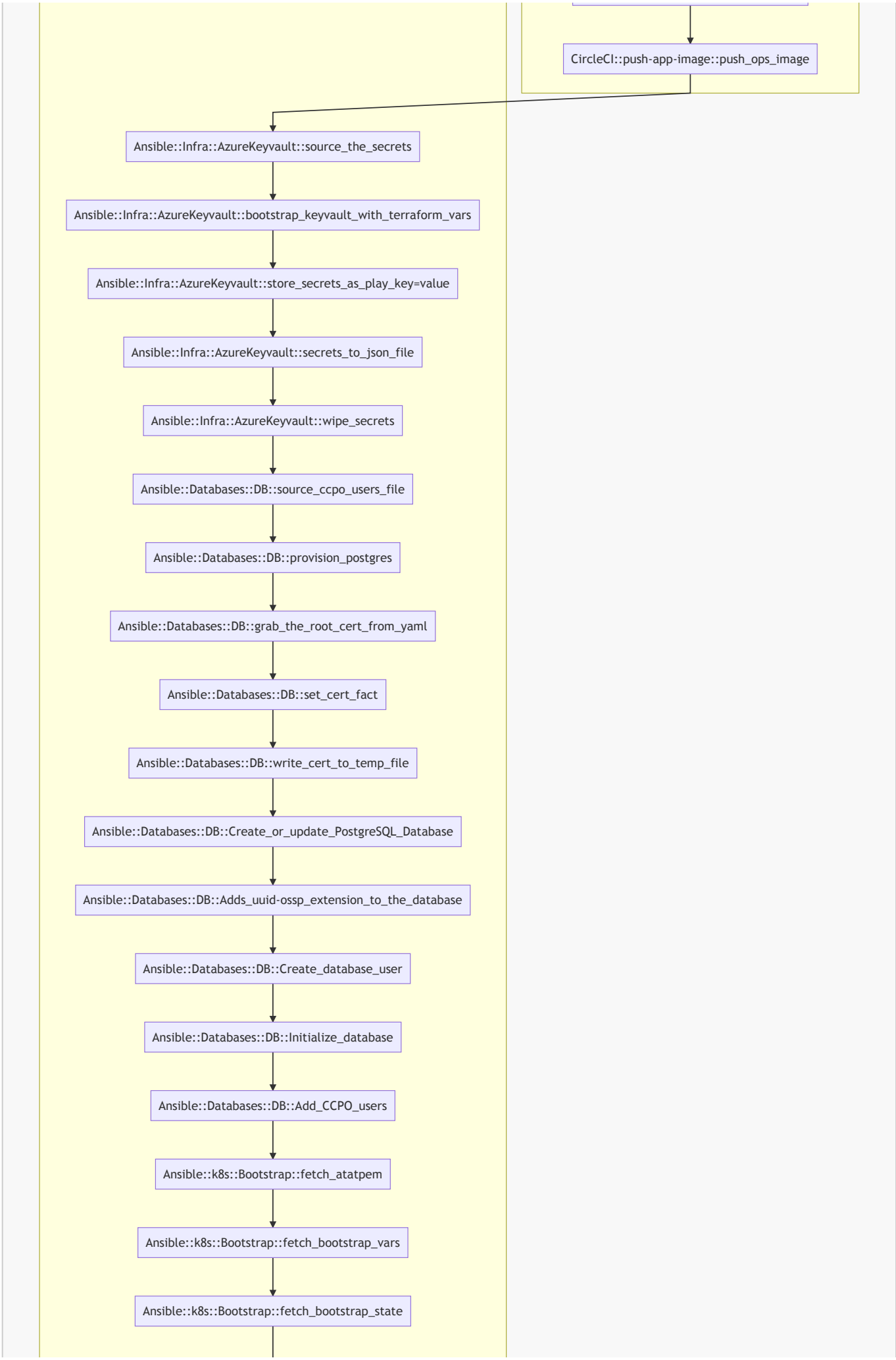
            - get bootstrap env outputs before destroy
            - destroy bootstrap env
                - tf destroy bootstrap env (no modules)
    - role: azure_keyvault
        - source the secrets
            - pickup application_env variables # set_fact
            - get app _output # terraform.get_backend.yml
            - include terraform/show_outputs.yml
            - set var values fact # set value
        - bootstrap keyvault with terraform vars
            - store deployment config and outputs values in **vault_name**
        - store secrets as plain key=value
            - Create secret
        - secrets to json file
            - set vault deploy tag # (set_fact)
            - get_secrets
                - get secrets
                - Include get_secrets
        - wipe secrets
        - get secrets
- databases.yml
    - role: db
        - source ccpo users file
            - Download CCPO Users file from storage account
        - provision postgres
            - grab the root cert from yaml
                - seems to just print the cert
            - set cert fact
            - write cert to temp file
            - Create (or update) PostgreSQL Database (azure resource - not in terraform?)
            - Adds uuid-ossp extension to the database
            - Create database user
            - Create database user
            - Initialize database
            - Add CCPO users
- k8s.yml
    - **if** private_cluster_deploy
        - role: bootstrap
            - **PARAMS**
                - show_bs_env = true
                - bootstrap_init_tf = true
                - deploy_bootstrap_env = false
                - fetch_bootstrap_state = true
            - fetch atat.pem
            - fetch bootstrap vars
            - fetch bootstrap state

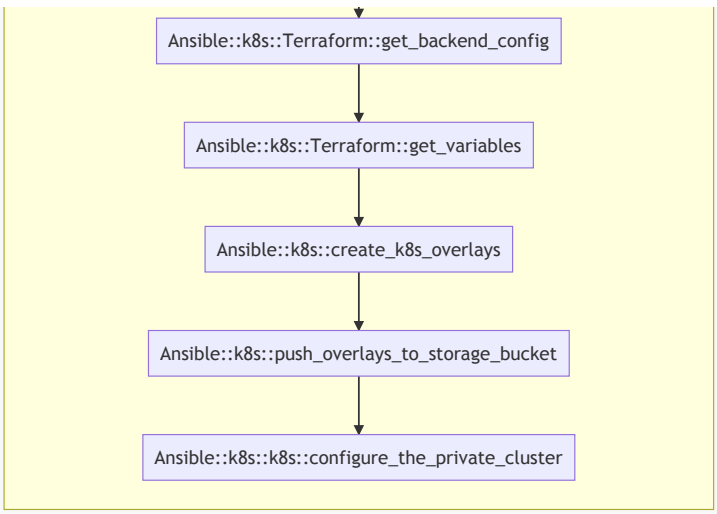
        - get vars
        - show outputs from run
    - role: terraform
        - **PARAMS**
            - show_app_env_outputs: true
            - deploy_app_env: false
            - init_tf: true
        - get backend config
        - get backend config
        - get variables
        - show outputs
- role: role: k8s
    - create k8s overlays
        - Create template output directory
        - Interpolate the templates
    - push overlays to storage bucket
        - Compress overlays
        - push overlays to a storage account
    - **if** private_cluster_deploy
        - configure the private cluster
            - Download k8s overlays
            - Extract private cluster configs
            - Connect to the {{ namespace }} kubernetes cluster
            - Create {{ namespace }} namespace
            - "Attach the ACR to the K8s cluster"
            - Assign Network Contributor role to the AKS Service Principal for the Virtual Network
            - Get vmss
            - Assign the Vault Reader identity to the AKS VMSS
            - Apply the storage class
            - Create kv namespace
            - Apply flex vol installer
            - Fetch latest `atat` image from {{ tf_app_env_outputs.container_registry_name.value }}
            - Extract latest tag
            - Fetch latest `nginx` image from {{ tf_app_env_outputs.container_registry_name.value }}
            - Extract latest nginx tag
            - Apply the rest of the Kubernetes config for the site
            - Obtain IP addresses
    - **else**
        - create k8s overlays
            - Create template output directory
            - Interpolate the templates
        - push overlays to storage bucket
            - Compress overlays

        - push overlays to a storage account
    - configure the cluster
        - Connect to the {{ namespace }} kubernetes cluster
        - Create {{ namespace }} namespace
        - "Attach the ACR to the K8s cluster"
        - Assign Network Contributor role to the AKS Service Principal for the Virtual Network
        - Get vmss
        - Assign the Vault Reader identity to the AKS VMSS
        - Apply the storage class
        - Create kv namespace
        - Apply flex vol installer
        - Fetch latest `atat` image from {{ tf_app_env_outputs.container_registry_name.value }}
        - Extract latest tag
        - Fetch latest `nginx` image from {{ tf_app_env_outputs.container_registry_name.value }}
        - Extract latest nginx tag
        - Apply the rest of the Kubernetes config for the site
        - Obtain IP addresses


circleci1
CircleCI::deploy-atat::checkout
CircleCI::deploy-atat::setup_remote_docker


CircleCI::deploy-atat::install_azure_cli
CircleCI::deploy-atat::log_into_ops_registry
CircleCI::deploy-atat::docker_build
CircleCI::deploy-atat::tag/push
CircleCI::deploy-atat::run_container_image
ContainerInstance1
Ansible::Infra::Crypto::gen_diffie_hellman
Ansible::Infra::Bootstrap::fetch_atatpem
Ansible::Infra::Bootstrap::fetch_bootstrap_vars
Ansible::Infra::Bootstrap::deploy_bootstrap_env
Ansible::Infra::Bootstrap::push_local_state_to_remote
circleci2
Ansible::Infra::CircleCi::run_circle_ci_ops_build
CircleCI::push-ops-image::docker_build
CircleCI::push-ops-image::push_ops_image
Ansible::Infra::CircleCi::Fetch_latest_ops_image_from
Ansible::Infra::Terraform::get_backend_config
Ansible::Infra::Terraform::deploy_application_env
Ansible::Infra::Terraform::tf_apply_no_modules
circleci3
Ansible::Infra::CircleCi::run_circle_ci_app_build
CircleCI::push-app-image::docker_build


CircleCI::push-app-image::push_ops_image
Ansible::Infra::AzureKeyvault::source_the_secrets
Ansible::Infra::AzureKeyvault::bootstrap_keyvault_with_terraform_vars
Ansible::Infra::AzureKeyvault::store_secrets_as_play_key=value
Ansible::Infra::AzureKeyvault::secrets_to_json_file
Ansible::Infra::AzureKeyvault::wipe_secrets
Ansible::Databases::DB::source_ccpo_users_file
Ansible::Databases::DB::provision_postgres
Ansible::Databases::DB::grab_the_root_cert_from_yaml
Ansible::Databases::DB::set_cert_fact
Ansible::Databases::DB::write_cert_to_temp_file
Ansible::Databases::DB::Create_or_update_PostgreSQL_Database
Ansible::Databases::DB::Adds_uuid-ossp_extension_to_the_database
Ansible::Databases::DB::Create_database_user
Ansible::Databases::DB::Initialize_database
Ansible::Databases::DB::Add_CCPO_users
Ansible::k8s::Bootstrap::fetch_atatpem
Ansible::k8s::Bootstrap::fetch_bootstrap_vars
Ansible::k8s::Bootstrap::fetch_bootstrap_state


Ansible::k8s::Terraform::get_backend_config
Ansible::k8s::Terraform::get_variables
Ansible::k8s::create_k8s_overlays
Ansible::k8s::push_overlays_to_storage_bucket
Ansible::k8s::k8s::configure_the_private_cluster