Okay, I understand. You have a developer-controlled enableOnlyInActiveRegion flag, but you want to override or augment its logic with the admin-controlled ActiveRegion variable group. The core idea is to introduce a new variable (let's call it finalEnabledState) that will ultimately determine the enabled property in your ARM template. This finalEnabledState will be set based on:

1. The developer's enableOnlyInActiveRegion flag.
2. The derived region from the alert rule name (e.g., WEU or NEU).
3. The admin's ActiveRegion variable from the variable group.

Here's how you can modify your Bash script in the pipeline to achieve this, fitting into your existing variable setting block:

Assumptions:

- You have an **admin-managed Variable Group** named DR-Configuration (or similar) with a variable ActiveRegion (e.g., WEU or NEU).
- Your pipeline *already links* this variable group using variables: - group: DR-Configuration.
- Developers are providing an alert.enableOnlyInActiveRegion value (true/false) in their input file (which you are currently setting to a pipeline variable enableOnlyInActiveRegion).
- Your alert rule names contain WEU or NEU for region-specific alerts.

Modified Variable Setting Part in your Pipeline (Bash Script):

```
# ... your existing variable setting
echo "##vso[task.setvariable variable=AlertRuleName]$(AlertRuleName-
QR-AZURE-$(env)-DIGITAL-${{ upper(parameters.podname) }}-${{
alert.AlertRuleName }})"
echo "##vso[task.setvariable variable=appname]$(appname)"
echo "##vso[task.setvariable variable=frequencyInMinutes]${{
alert.frequencyInMinutes }}"
echo "##vso[task.setvariable variable=timeWindowInMinutes]${{
alert.timeWindowInMinutes }}"
echo "##vso[task.setvariable variable=severity]${{ alert.severity }}"
echo "##vso[task.setvariable variable=operator]${{ alert.operator }}"
echo "##vso[task.setvariable variable=threshold]${{ alert.threshold
}}"
echo "##vso[task.setvariable variable=autoMitigate]${{
alert.autoMitigate }}"

# Capture developer's intended 'enabled' state for the alert (from the
input file)
echo "##vso[task.setvariable variable=devEnabled]${{ alert.enabled }}"

# Capture developer's 'enableOnlyInActiveRegion' flag
# This variable's value from input file could be 'true' or 'false'
developerEnableOnlyInActiveRegion="${{ alert.enableOnlyInActiveRegion
}}"
echo "##vso[task.setvariable variable=enableOnlyInActiveRegion]${{
alert.enableOnlyInActiveRegion }}"


# --- NEW LOGIC STARTS HERE ---

# 1. Get the current active region from the admin-managed variable
group
```

```
ADMIN_ACTIVE_REGION="$(ActiveRegion)" # This fetches from the linked
variable group
echo "Admin-defined active region: $ADMIN_ACTIVE_REGION"

# 2. Derive the primary region for the current alert rule from its
name
ALERT_NAME_LOWERCASE=$(echo "$(AlertRuleName)" | tr '[:upper:]'
'[:lower:]') # Convert alert name to lowercase
DERIVED_ALERT_PRIMARY_REGION=""

if [[ "$ALERT_NAME_LOWERCASE" == *weu* ]]; then
    DERIVED_ALERT_PRIMARY_REGION="WEU"
elif [[ "$ALERT_NAME_LOWERCASE" == *neu* ]]; then
    DERIVED_ALERT_PRIMARY_REGION="NEU"
fi

echo "Derived primary region for alert '$(AlertRuleName)':
$DERIVED_ALERT_PRIMARY_REGION"


# 3. Determine the FINAL 'enabled' state for the ARM template
FINAL_ENABLED_STATE="true" # Default to enabled

# Condition 1: Is this alert meant to be region-specific by the
developer?
# Condition 2: Could we successfully derive a region from the alert's
name?
if [[ "$developerEnableOnlyInActiveRegion" == "true" && -n
"$DERIVED_ALERT_PRIMARY_REGION" ]]; then
    # This is a region-specific alert, and we know its primary region
from the name
    if [[ "$DERIVED_ALERT_PRIMARY_REGION" == "$ADMIN_ACTIVE_REGION"
]]; then
        FINAL_ENABLED_STATE="true" # Enable if it's the active region
    else
        FINAL_ENABLED_STATE="false" # Disable if it's the passive
region
    fi
else
    # This alert is either NOT region-specific (developer set flag to
false)
    # OR we couldn't derive its primary region from the name.
    # In these cases, it should always be enabled, regardless of the
active region.
    FINAL_ENABLED_STATE="${{ alert.enabled }}" # Revert to developer's
explicit 'enabled' flag if not region-specific
                                              # Or default to true as
per your current logic for non-regional alerts
                                              # If alert.enabled is not
always present, you might want a default.
                                              # E.g., if [[ -z "${{
alert.enabled }}" ]]; then FINAL_ENABLED_STATE="true"; else
```

```
FINAL_ENABLED_STATE="${{ alert.enabled }}"; fi
fi

echo "Final calculated enabled state for alert '$(AlertRuleName)':
$FINAL_ENABLED_STATE"
echo "##vso[task.setvariable variable=enabled]$FINAL_ENABLED_STATE" #
This variable will be used in the ARM template

# --- NEW LOGIC ENDS HERE ---

# Your original logic:
# enableOnlyInActiveRegion=$(enableOnlyInActiveRegion) # This line
becomes redundant as we processed it above
# if [ -z "$enableOnlyInActiveRegion" ]; then
#   enableOnlyInActiveRegion="False"
# fi
# echo "##vso[task.setvariable
variable=enableOnlyInActiveRegion]$enableOnlyInActiveRegion"

# Your next task, `Resolve_variable`, will now pick up the
`$(enabled)` variable,
# which holds `FINAL_ENABLED_STATE`.

# ... rest of your pipeline tasks
```

Explanation of Changes:

1. **ADMIN_ACTIVE_REGION="$(ActiveRegion)"**: This line directly retrieves the value of the ActiveRegion variable from the linked DR-Configuration variable group. This is the crucial administrative override.
2. **Deriving DERIVED_ALERT_PRIMARY_REGION**:
   - It takes the AlertRuleName (which you've already set in your pipeline).
   - tr '[:upper:]' '[:lower:]' converts the name to lowercase for robust string matching.
   - if [["$ALERT_NAME_LOWERCASE" == *weu*]] and elif [["$ALERT_NAME_LOWERCASE" == *neu*]] checks if the alert name contains "weu" or "neu". This assumes your naming standard is consistent.
   - DERIVED_ALERT_PRIMARY_REGION will be set to WEU, NEU, or remain empty if neither is found.
3. **Determining FINAL_ENABLED_STATE**:
   - It checks two conditions to decide if the "active region" logic should apply:
     - $developerEnableOnlyInActiveRegion == "true": Is the developer explicitly flagging this alert as region-specific?
     - -n "$DERIVED_ALERT_PRIMARY_REGION": Were we able to successfully identify a primary region for this alert from its name?
   - **If BOTH are true**: This is where the core logic applies. FINAL_ENABLED_STATE is set to "true" only if the DERIVED_ALERT_PRIMARY_REGION matches the ADMIN_ACTIVE_REGION. Otherwise, it's "false".
   - **If EITHER is false**: (Meaning the developer didn't flag it as region-specific OR we couldn't determine its region from the name). In this scenario, we assume it's a global alert or one that should always be enabled, so we revert to the developer's original alert.enabled value. If alert.enabled might not always be present, you might want to default it to "true" explicitly.

4. **echo "##vso[task.setvariable variable=enabled]$FINAL_ENABLED_STATE"**: This is the critical line. It sets the pipeline variable enabled (which your ARM template consumes) to the FINAL_ENABLED_STATE determined by your logic.

How your ARM template consumes this:
Your ARM template should be parameterized to accept the enabled variable directly, like this:

```
"properties": {
  "enabled": "[parameters('enabled')]", // This parameter in ARM will
be passed by your pipeline
  // ... rest of properties
},
"tags": {
  "RegionSpecificAlert":
"[string(parameters('isRegionSpecificAlert'))]", // Developers set
this
  "PrimaryRegion": "[variables('derivedPrimaryRegion')]" // Derived in
ARM from name
}
```

And in your pipeline's az deployment group create command, you'd pass it:

```
az deployment group create \
  --resource-group "$RESOURCE_GROUP" \
  --template-file "$TEMPLATE_FILE" \
  --parameters \
    alertRuleName="$ALERT_NAME" \
    alertRuleLocation="$ALERT_LOCATION" \
    isRegionSpecificAlert=$IS_REGION_SPECIFIC \ # Developer's flag
    enabled="$FINAL_ENABLED_STATE" \           # The critical,
calculated variable
    activeDeploymentRegion="$(ActiveRegion)" \ # Pass admin-controlled
region to ARM template for consistency/tagging
    alertQuery="$ALERT_QUERY" \
    alertThreshold="$ALERT_THRESHOLD" \
    --output none
```

Developer's Responsibility:

- **Naming Convention:** Continue to use WEU or NEU in the alert rule name for region-specific alerts.
- **isRegionSpecificAlert Flag:** For an alert to be managed by the active/passive region logic, the developer *must* set isRegionSpecificAlert (or whatever your flag is named in the input file) to true. If this flag is false, the alert will always be deployed with its developer-intended enabled state (likely true).

This approach gives you the administrative control you desire, while leveraging existing developer practices and naming conventions.