Grafana

Create Grafana Dashboard to Monitor Skybox/TF Version and Tenant Status

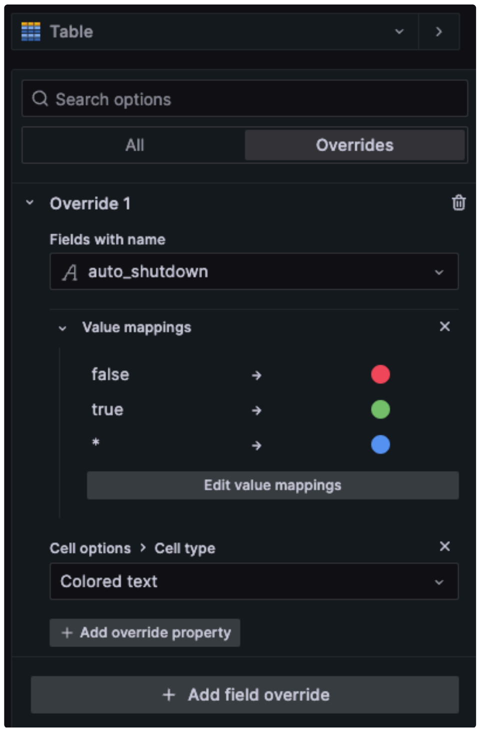
Objective:

- Build a pipeline to collect and store information about tenants in a multi-tenant AWS environment.
- Visualize the collected data in a Grafana dashboard.

Components:

1. Pipeline **-** aws_mt_inventory
   a. Get mapping of account_id-account_name from AWS - - using `aws organizations list-accounts | jq -r '.Accounts[]'` with root access.
   b. Get a list of all multi-tenant from consul
   c. Sub-pipeline - aws_mt_account_inventory - runs in parallel for all accounts
      i. Connect to AWS to get the stateful-sets of the tenants of the account
      ii. Script (aws_mt_inventory.py) - collects all the data needed and send to elastic search with a unique id per each tenant.
   d. Pipeline - aws_eks_tenant_provisioning - already exists, just add an upsert to a document, to ensure consistent data from multiple sources.
      i. if tenant is destroyed - change its EKS status to deleted.
2. **Elasticsearch Document:**
   - **Fields:**
     - customer (string): Name of the customer.
     - ou (string): Organizational Unit.
     - account_id (string): AWS account ID.
     - aws_account_name (string): Name of the AWS account.
     - terraform_version (string): Terraform version used for provisioning.
     - connected_to_voltra (boolean): Indicates connection to Voltra.
     - skybox_version (string): Skybox version (tenant and consul tag).
     - is_url_reachable (boolean): Indicates if the URL is reachable.
     - stateful_set_status (string): Status of the stateful set.
     - auto_shutdown (boolean)
     - ebs_size (string)
     - enable_cloudwatch_agent (boolean)
     - enable_datadog_agent (boolean)
     - env (string)
     - size (string)
     - skybox_public_url (string)
     - tag (string)
3. **Grafana Dashboard (AWS Multi-Tenant Summary):**
   - **Data Source:** Elasticsearch indices containing "aws_mt_inventory*" data.
   - **Query:** Filters data based on customer, OU, account ID, and account name.
     - query: customer: $customer AND ou: $ou AND account_id: $account_id AND aws_account_name: $aws_account_name AND _index:"aws_mt_inventory
     - *index:"aws_mt_inventory*_ownership
   - **Transformations:**
     - Group by transformation -
       - Groups data by customer (primary grouping).
       - Calculates "first*" for each field to display the latest non-null value per customer.
       - Ignored for column not to be displayed.
     - Organize fields transformation - organizes fields with desired column names + order columns.
   - **Display:**
     - Table format.

- `consul_client.put(f"{AWS_OU_PREFIX}/{ou}/{AWS_MT_CONFIGURATION_SUFFIX}/{account}/{customer}/{MT_ACCOUNTS_AUTO_SHUTDOWN_SUFFIX}", custom_control_value)`
- Hides unnecessary columns.
- Applies color coding to specific columns based on value patterns.
  - add override property → value mapping (does not exist for this client, ^(?!true|false).*$)
  - add override property → cell options > cell type



Table

Search options

All | Overrides

Override 1

Fields with name: auto_shutdown

Value mappings

false	→	🔴
true	→	🟢
*	→	🔵

Edit value mappings

Cell options > Cell type: Colored text

Add override property

Add field override

Update doc in elasticsearch to increase Grafana dashboard capabilities

AIM: take the current index and add parameters to it to create a more robust Grafana dashboard.

Step 1: Search for Usage of `junit_parser` in Code

1. Open **PyCharm**.
2. Use the shortcut `Command + Shift + F` to search for `junit_parser` across all files.
3. Update all references where documents are pushed to Elasticsearch through it:
   - `pipeline = source to target + NightlyRun`
   - Replace **index name** in `parse_results` with:

     ```
     es_api.bulk_insert("junit_yalmaliah_test", docs, elastic_password=elastic_password)
     ```

Step 2: Modify `junit_parser.py`

1. **Add Arguments**:
   - Add short and long argument names.
   - Follow the function call trail to ensure proper integration.
2. **Update Index**: Ensure the target index name (`junit_yalmaliah_test`) is used in the `parse_results` function.

Step 3: Integrate Updates into the Pipeline

1. **Add Arguments in the Pipeline**:
   - Through the script (`sh`): Add arguments as-is.
   - Through the pipeline config: Add arguments as environment variables (`env.VAR_NAME`).
2. **Find Environment Variables**:
   - Open a pipeline job in Jenkins, view as **plain text**, and check all available variables.
   - Alternatively - view environment variables: https://jenkins-git/env-vars.html.

3. **Push Changes to Git**:
   - Create a branch in Skybox named after the Jira ticket (e.g., `SKY-272148`).
   - In `skybox/server/parent/pom.xml`, add a comment to trigger the pipeline.
   - Submit a merge request.

Step 4: Elasticsearch Setup

1. Access Kibana: https://sb-kibana. password from secert server - sb-elasticsearc
2. Add the testing index:
   - Go to **Management → Dev Tools → Console**.
   - Use the following command to create the index:

     `PUT junit_yalmaliah_test`
   - Use the following command to delete the index (needs to be recreated for each run of the pipeline):
     `DELETE junit_yalmaliah_test`
3. Confirm the document format:

   `GET junit_yalmaliah_test/_search`

Step 5: Update Grafana Dashboard

1. Access Grafana: https://sb-grafana. password from secert server - Devopsadmin **or** my user (AD)
2. **Duplicate the Dashboard**: Never edit the original.
3. **Add New Variables**:
   - Go to the variable bar → **Settings → Variables**.
4. **Update Queries**:
   - Edit components → Add `AND x: $x` to the query.

Step 6: Troubleshooting

- Debug script arguments, by adding to `junit_parser`:

```
print("Parsed options:", opts)
print("Remaining arguments:", args)
```

- Add timestamps to pipeline if needed:

```
script { def currentTime = sh(script: "date '+%H:%M:%S'", returnStdout: true).trim() echo "Timestamp: ${currentTime}" sh label:
'', script: """ python3 pipelineUtils/junit_parser.py -m ${gitlabMergeRequestIid} -t ${gitlab_api} \ -p ${WORKSPACE}/skyboxview/
-u ${db_user} -x ${db_pass} -s ${gitlabSourceBranch} \ -d ${gitlabTargetBranch} -o ${gitlabUserEmail} -e ${elastic_pass} -i
${BUILD_ID} \ -r ${gitlabMergeRequestIid} -j ${JOB_NAME} -a ${currentTime} """ }
```