Milestone 1:

Github Repo Link

https://github.com/theashwin/ml4se

Team Members


```
Ashwin Patil <anpatil2@illinois.edu>
Chinmay Saraf <csaraf2@illinois.edu>
Kedar Takwane <takwane2@illinois.edu>
Maahi Patel <maahidp2@illinois.edu>
```


Dataset used for Code Analysis:

- We have used CodeSearchNet dataset

Python

- As CodeQL needs .py file for python based code analysis, we extracted random 100 functions from CodeSearchNet dataset using code from notebook
- The json format of datapoints can be found in python-data
- The folder has all .py files considered for the analysis.

Java

- As CodeQL needs java project for java code analysis, we have imported hazelcast repository which is one of the repository used by CodeSearchNet in its dataset.
- We have considered some java files from hazelcast in the code analysis. The folder has all .java files considered for the analysis.
- The json format of datapoints can be found in java-data

CodeQL Queries and Python scripts

Python

- Code for Data flow: python-data-flow
- Code for Control flow: python-control-flow

Java

- Code for Data flow: java-data-flow
- Code for Control flow: java-control-flow

Instructions about how to use the code:

Setup CodeQL CLI and Visual Studio Extension

1. Install CodeQL CLI
2. Install VS Code extension and setting it up - Follow all steps mentioned in this link. Please check if the queries are running with given example database

Add Database

1. Open VS Code
2. Open CodeQL Window (click QL icon from left panel)
3. Hover on Databases tab - it'll show multiple icons to add database

   a. From Archive
   b. From Folder
   c. Download Database
   d. From Github
4. For Java, CodeQL needs whole Java Project. So, for this milestone we have used hazelcast Java project. Add database in CodeQL using **From Github** option. The VS Code window may pop out to selcting language. Select Java language.
5. For Python, CodeQL needs files with .py extension. For this milestone, we have added data in folder. Add this database in CodeQL from **From Folder** option. The VS Code window may pop out to selcting language. Select Python language.
6. Under Databases tab in VS Code, hover on databases and click on **Set Current Database** to query respective database.

Execute Queries

On VS Code's CodeQL window, through **Command Palette**, open **CodeQL: Quick Query**. This will open `quick-query.ql` file.

Java

- Control Flow -
  a. Copy content from control_flow.ql and paste it in `quick-query.ql` .
  b. Right Click on Editor and select **CodeQL: Run Query on Selected Database**.
- Data Flow -
  a. Copy content from data_flow.ql and paste it in `quick-query.ql` .
  b. Right Click on Editor and select **CodeQL: Run Query on Selected Database**.

Python

- Control Flow -
  a. Copy multiple query files from control_flow and paste it in `quick-query.ql` and run individually.
  b. Right Click on Editor and select **CodeQL: Run Query on Selected Database**.
  c. These queries will extract different parts of code for CFG generation. d. Code for this is given in python.ipynb
- Data Flow -
  a. Copy multiple query files from data_flow and paste it in `quick-query.ql` and run individually.
  b. Right Click on Editor and select **CodeQL: Run Query on Selected Database**.
  c. These queries will extract different parts of code for DFG generation. d. Code for this is given in python.ipynb

Results

Python

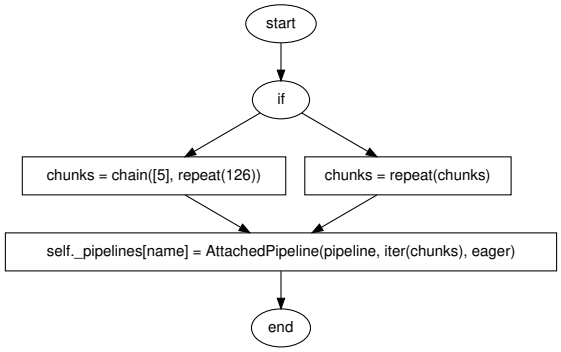
To map python code snippet with the Control Flow and Data Flow Graphs: For each `n.py` file from folder, check control-flow and data-flow with name `n.svg`

- Code Snippet #1

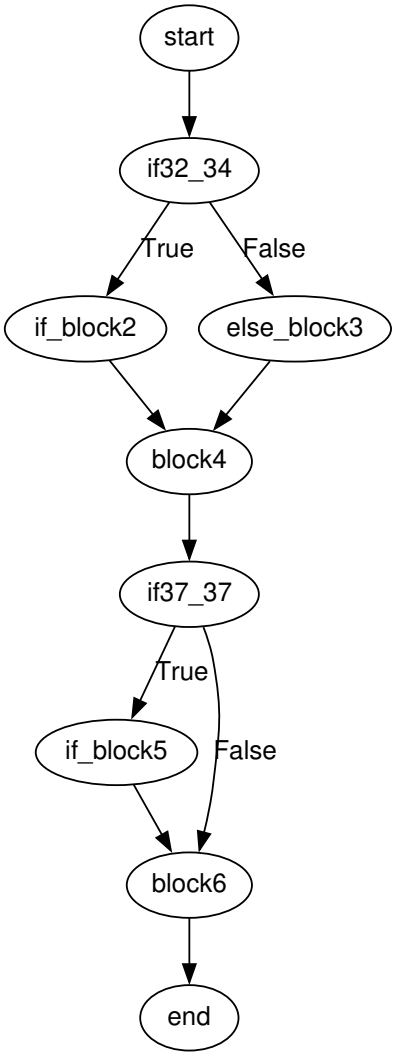
    1. Code Snippet

```
def attach_pipeline(self, pipeline, name, chunks=None, eager=True):
    """Register a pipeline to be computed at the start of each day.

    Parameters
    ----------
    pipeline : Pipeline
        The pipeline to have computed.
    name : str
        The name of the pipeline.
    chunks : int or iterator, optional
        The number of days to compute pipeline results for. Increasing
        this number will make it longer to get the first results but
        may improve the total runtime of the simulation. If an iterator
        is passed, we will run in chunks based on values of the iterator.
        Default is True.
    eager : bool, optional
        Whether or not to compute this pipeline prior to
        before_trading_start.

    Returns
    -------
    pipeline : Pipeline
        Returns the pipeline that was attached unchanged.

    See Also
    --------
    :func:`zipline.api.pipeline_output`
    """
    if chunks is None:
        # Make the first chunk smaller to get more immediate results:
        # (one week, then every half year)
        chunks = chain([5], repeat(126))
    elif isinstance(chunks, int):
        chunks = repeat(chunks)

    if name in self._pipelines:
        raise DuplicatePipelineName(name=name)

    self._pipelines[name] = AttachedPipeline(pipeline, iter(chunks), eager)

    # Return the pipeline to allow expressions like
    # p = attach_pipeline(Pipeline(), 'name')
    return pipeline
```

2. Data Flow Graph -



3. Control Flow Graph -

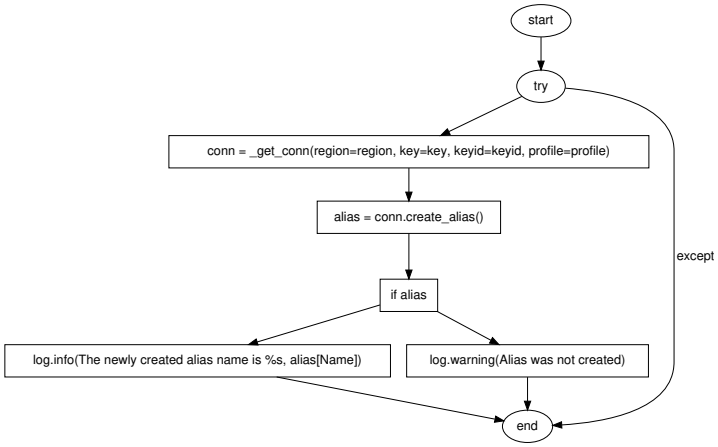


- Code Snippet #2

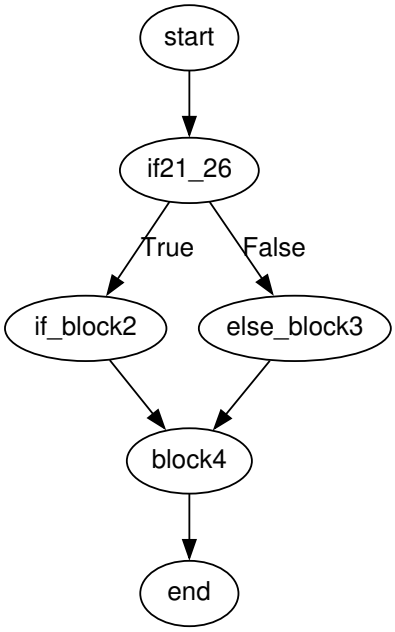
    1. Code Snippet

```
def create_alias(FunctionName, Name, FunctionVersion, Description="",
           region=None, key=None, keyid=None, profile=None):
 '''
 Given a valid config, create an alias to a function.

 Returns {created: true} if the alias was created and returns
 {created: False} if the alias was not created.

 CLI Example:

 .. code-block:: bash

  salt myminion boto_lamba.create_alias my_function my_alias $LATEST "An alias"

 '''
 try:
  conn = _get_conn(region=region, key=key, keyid=keyid, profile=profile)
  alias = conn.create_alias(FunctionName=FunctionName, Name=Name,
                            FunctionVersion=FunctionVersion, Description=Description)
  if alias:
      log.info('The newly created alias name is %s', alias['Name'])

      return {'created': True, 'name': alias['Name']}
  else:
      log.warning('Alias was not created')
      return {'created': False}
 except ClientError as e:
  return {'created': False, 'error': __utils__['boto3.get_error'](e)}
```

    2. Data Flow Graph -



3. Control Flow Graph -



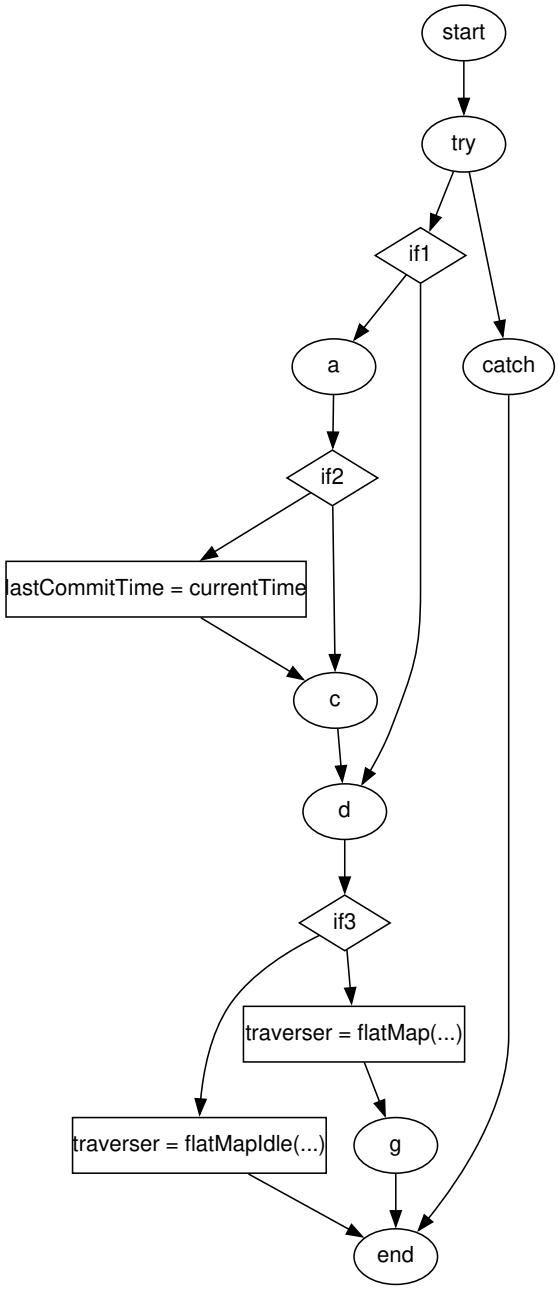
Java

To map java code snippet with the Control Flow and Data Flow Graphs: For each `image_name.svg` from control-flow-graphs or data-flow-graphs, please refer to file java.json. In this file, check for json_object with value of `label` as `image_name`. The json object has details about **function name, link to file containing the function, and code**.

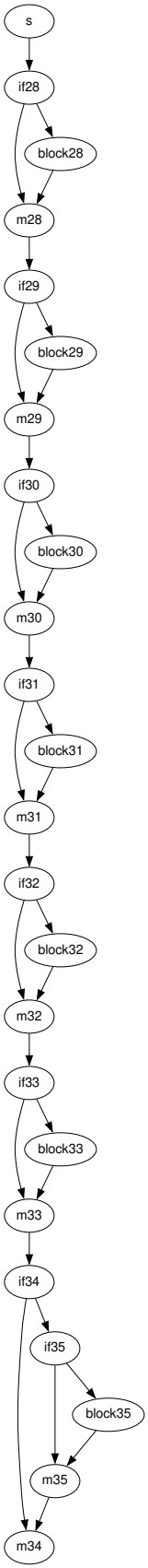
- Code Snippet #1
    1. Code Snippet

```
public boolean complete() {
    if (!emitFromTraverser(traverser)) {
        return false;
    }
    if (reconnectTracker.needsToWait()) {
        return false;
    }
    if (!isConnectionUp()) {
        return false;
    }
    if (snapshotInProgress) {
        return false;
    }

    try {
        if (!snapshotting && commitPeriod > 0) {
            long currentTime = System.nanoTime();
            if (currentTime - lastCommitTime > commitPeriod) {
                task.commit();
                lastCommitTime = currentTime;
            }
        }

        List<SourceRecord> records = task.poll();
        if (records == null || records.isEmpty()) {
            traverser = eventTimeMapper.flatMapIdle();
            emitFromTraverser(traverser);
            return false;
        }

        for (SourceRecord record : records) {
            Map<String, ?> partition = record.sourcePartition();
            Map<String, ?> offset = record.sourceOffset();
            state.setOffset(partition, offset);
            task.commitRecord(record, null);
        }

        if (!snapshotting && commitPeriod == 0) {
            task.commit();
        }

        traverser = Traversers.traverseIterable(records)
                .flatMap(record -> {
                    T t = map(record);
                    return t == null ? Traversers.empty() :
                            eventTimeMapper.flatMapEvent(t, 0, extractTimestamp(record));
                });
        emitFromTraverser(traverser);
    } catch (InterruptedException ie) {
        logger.warning("Interrupted while waiting for data");
        Thread.currentThread().interrupt();
    } catch (RuntimeException re) {
        reconnect(re);
    }

    return false;
}
```

2. Data Flow Graph -



3. Control Flow Graph -

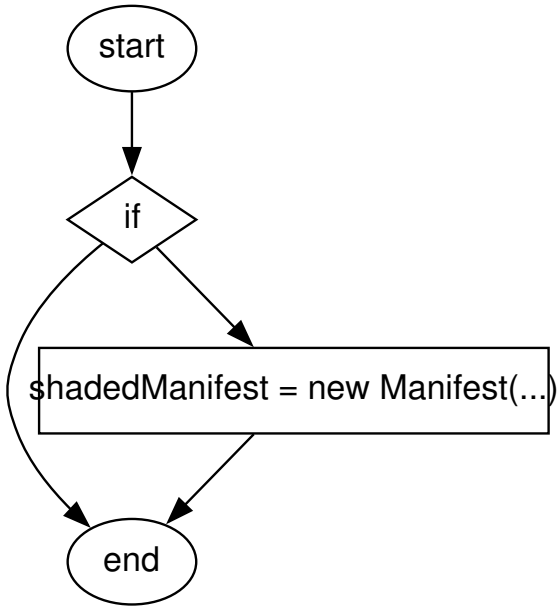


- Code Snippet #2

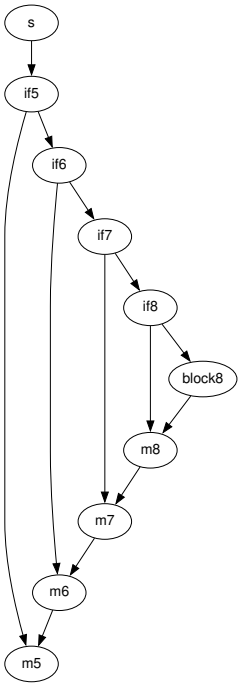
    1. Code Snippet

```
public void modifyOutputStream(JarOutputStream jarOutputStream) throws IOException {
    if (shadedManifest == null) {
        shadedManifest = new Manifest();
    }

    Attributes attributes = shadedManifest.getMainAttributes();

    if (overrideInstructions != null) {
        precompileOverrideInstructions();
        attributes.putValue(IMPORT_PACKAGE, join(shadeImports().iterator(), ","));
        attributes.putValue(EXPORT_PACKAGE, join(shadeExports().iterator(), ","));
    }

    attributes.putValue("Created-By", "HazelcastManifestTransformer through Shade Plugin");

    if (mainClass != null) {
        attributes.put(Attributes.Name.MAIN_CLASS, mainClass);
    }

    if (manifestEntries != null) {
        for (Map.Entry<String, Object> entry : manifestEntries.entrySet()) {
            attributes.put(new Attributes.Name(entry.getKey()), entry.getValue());
        }
    }

    // the Manifest in hazelcast uberjar won't have the Automatic-Module-Name
    attributes.remove(AUTOMATIC_MODULE_NAME);

    jarOutputStream.putNextEntry(new JarEntry(JarFile.MANIFEST_NAME));
    shadedManifest.write(jarOutputStream);
    jarOutputStream.flush();
}
```

2. Data Flow Graph -



3. Control Flow Graph -


s
if5
if6
if7
if8
block8
m8
m7
m6
m5