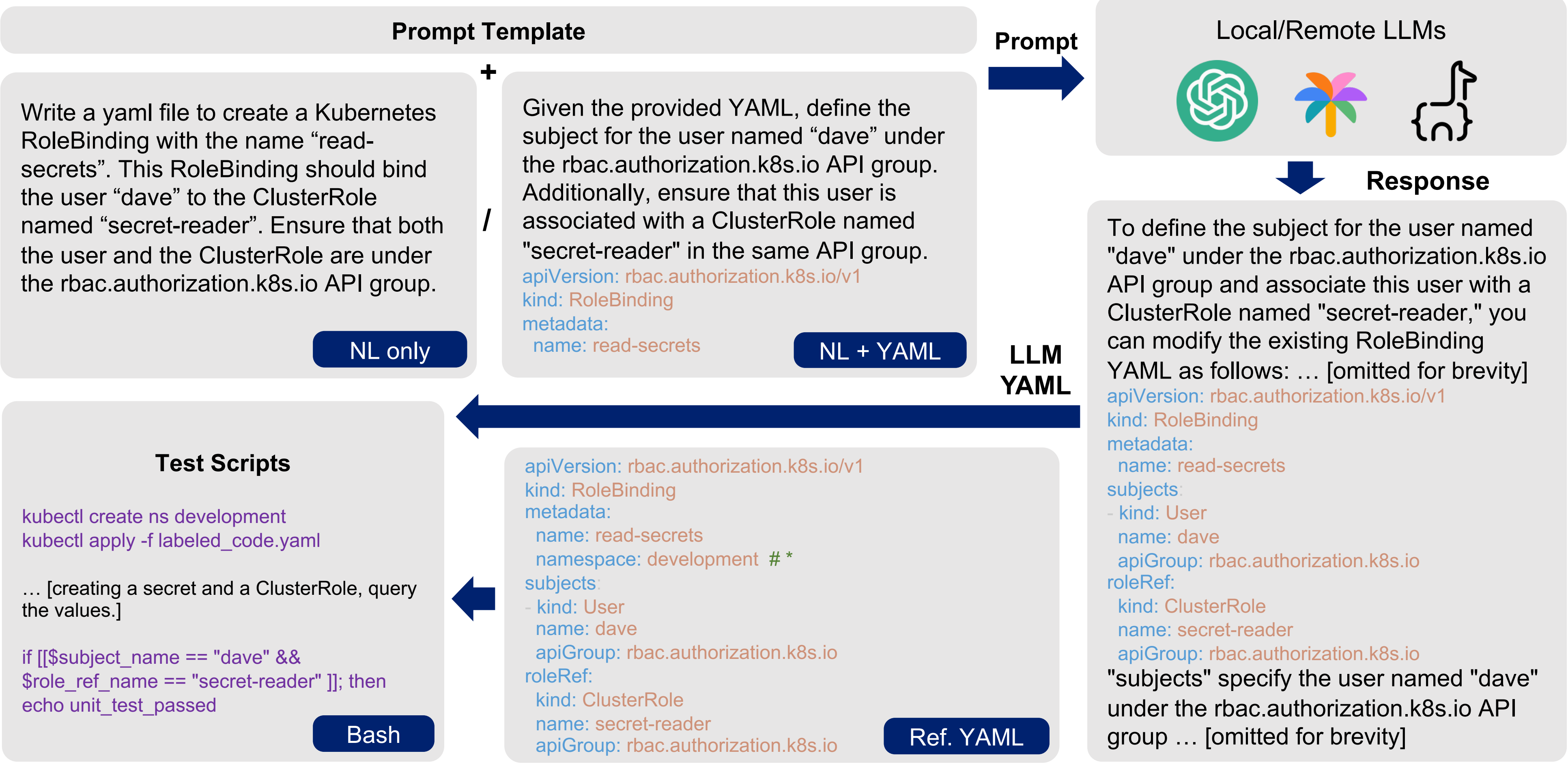

Write a yaml file to create a Kubernetes RoleBinding with the name “read-secrets”. This RoleBinding should bind the user “dave” to the ClusterRole named “secret-reader”. Ensure that both the user and the ClusterRole are under the rbac.authorization.k8s.io API group.

Given the provided YAML, define the subject for the user named “dave” under the rbac.authorization.k8s.io API group. Additionally, ensure that this user is associated with a ClusterRole named "secret-reader" in the same API group.

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-secrets
```

To define the subject for the user named "dave" under the rbac.authorization.k8s.io API group and associate this user with a ClusterRole named "secret-reader," you can modify the existing RoleBinding YAML as follows: … [omitted for brevity]

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-secrets
subjects:
- kind: User
  name: dave
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

"subjects" specify the user named "dave" under the rbac.authorization.k8s.io API group … [omitted for brevity]

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-secrets
  namespace: development  # *
subjects:
- kind: User
  name: dave
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

Test Scripts

```bash
kubectl create ns development
kubectl apply -f labeled_code.yaml

… [creating a secret and a ClusterRole, query the values.]

if [[$subject_name == "dave" &&
$role_ref_name == "secret-reader" ]]; then
echo unit_test_passed
```