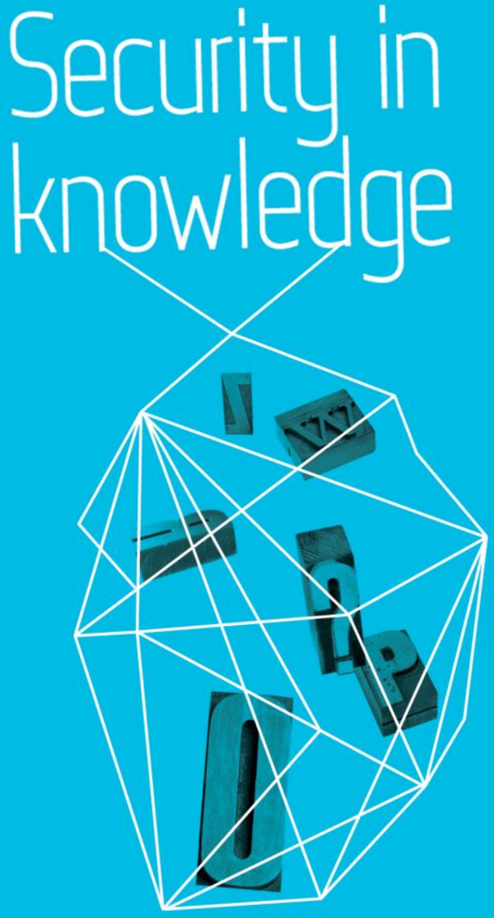
RSA CONFERENCE 2013



Integrating OpenStack's Keystone Service with an Access Management System

Suresh Kumar

EMC Solution Group, EMC Corporation
suresh.kumar4@emc.com

Robert Polansky

RSA, The Security Division of EMC
robert.polansky@rsa.com

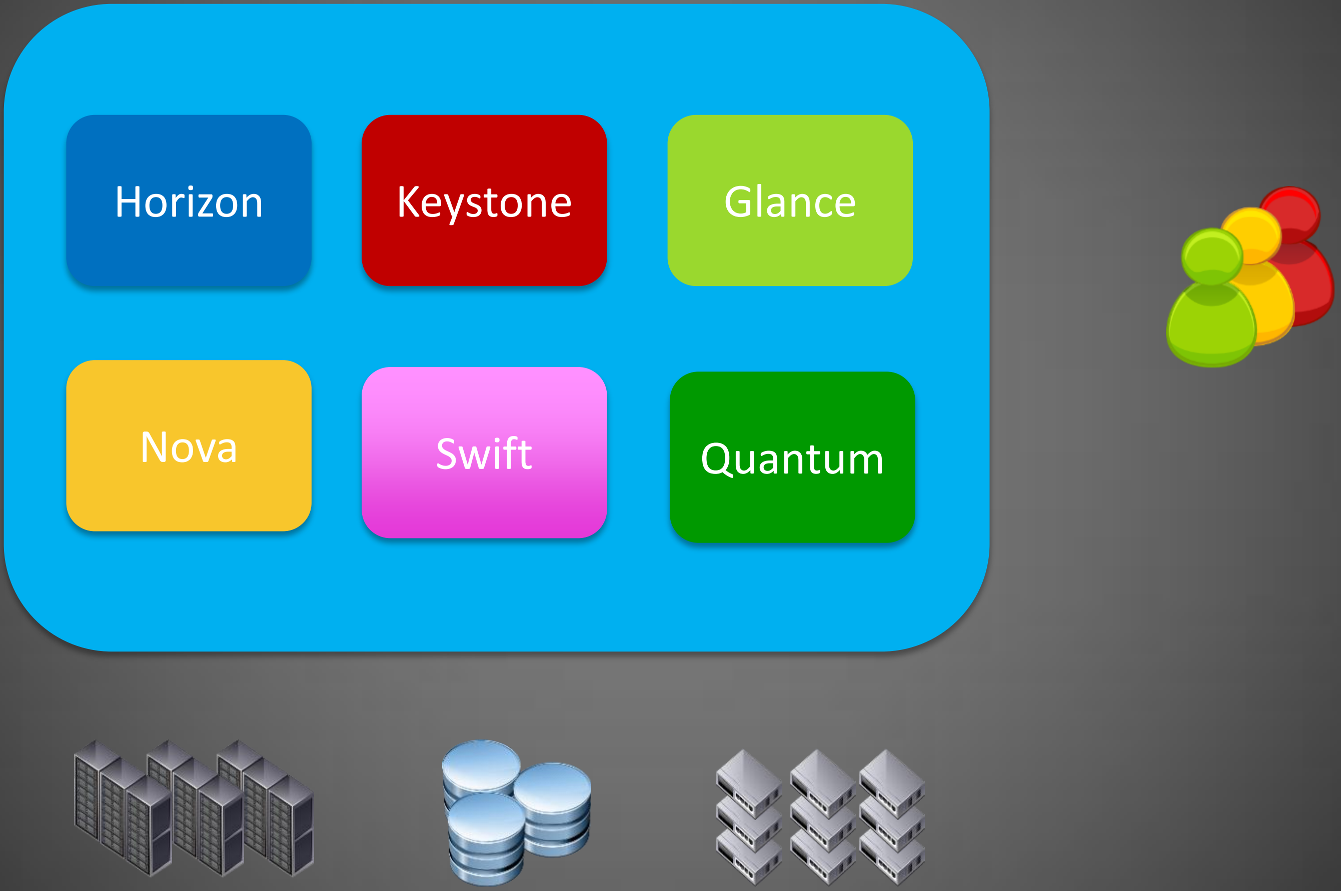
Session ID: IAM-F43

Session Classification: Intermediate

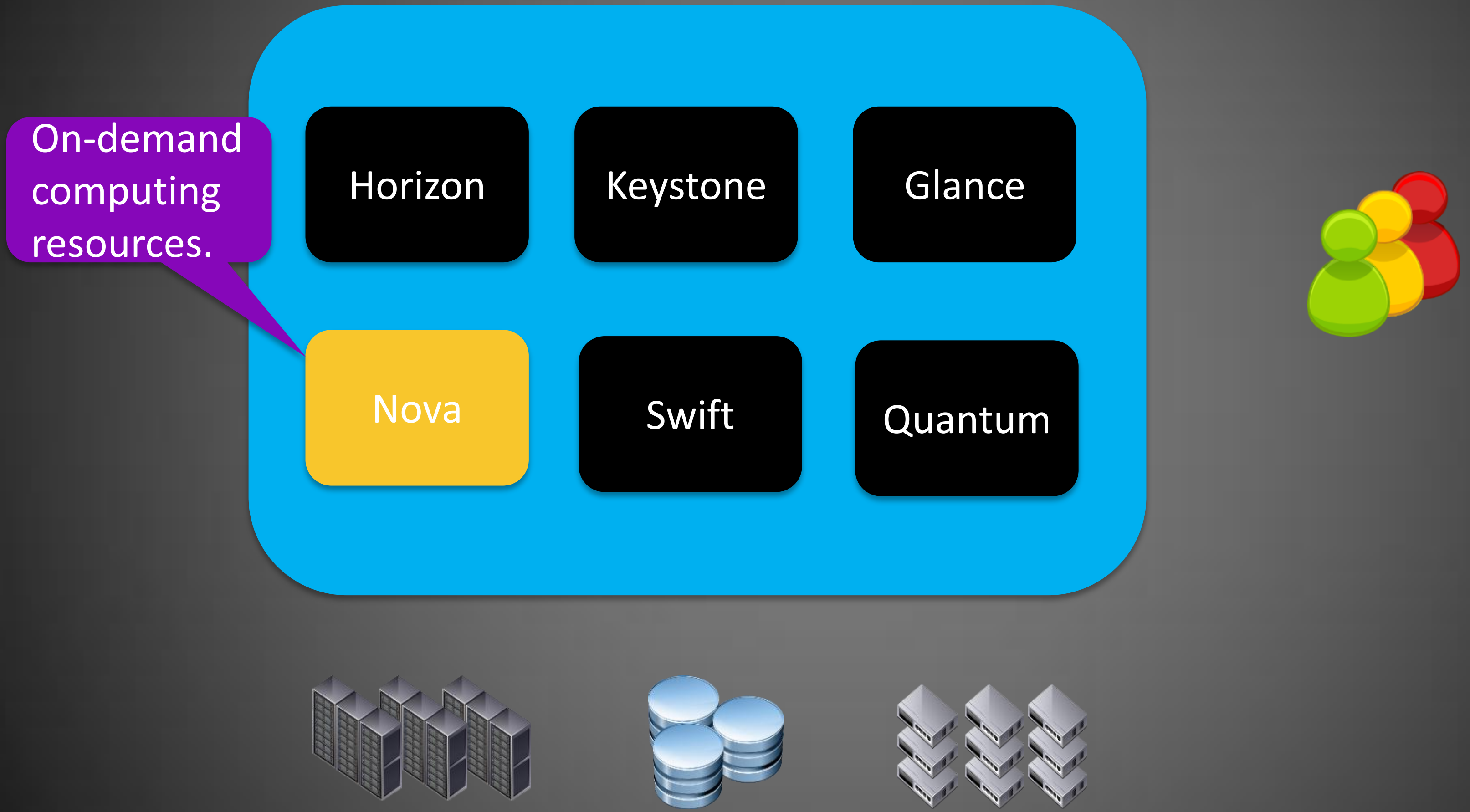
Why are we here?

- OpenStack's momentum in the marketplace.
- Discuss OpenStack's IAM implementation with RSA Access Manager.
- Integration challenges and solutions.
- Explore other avenues of integration.

OpenStack Components



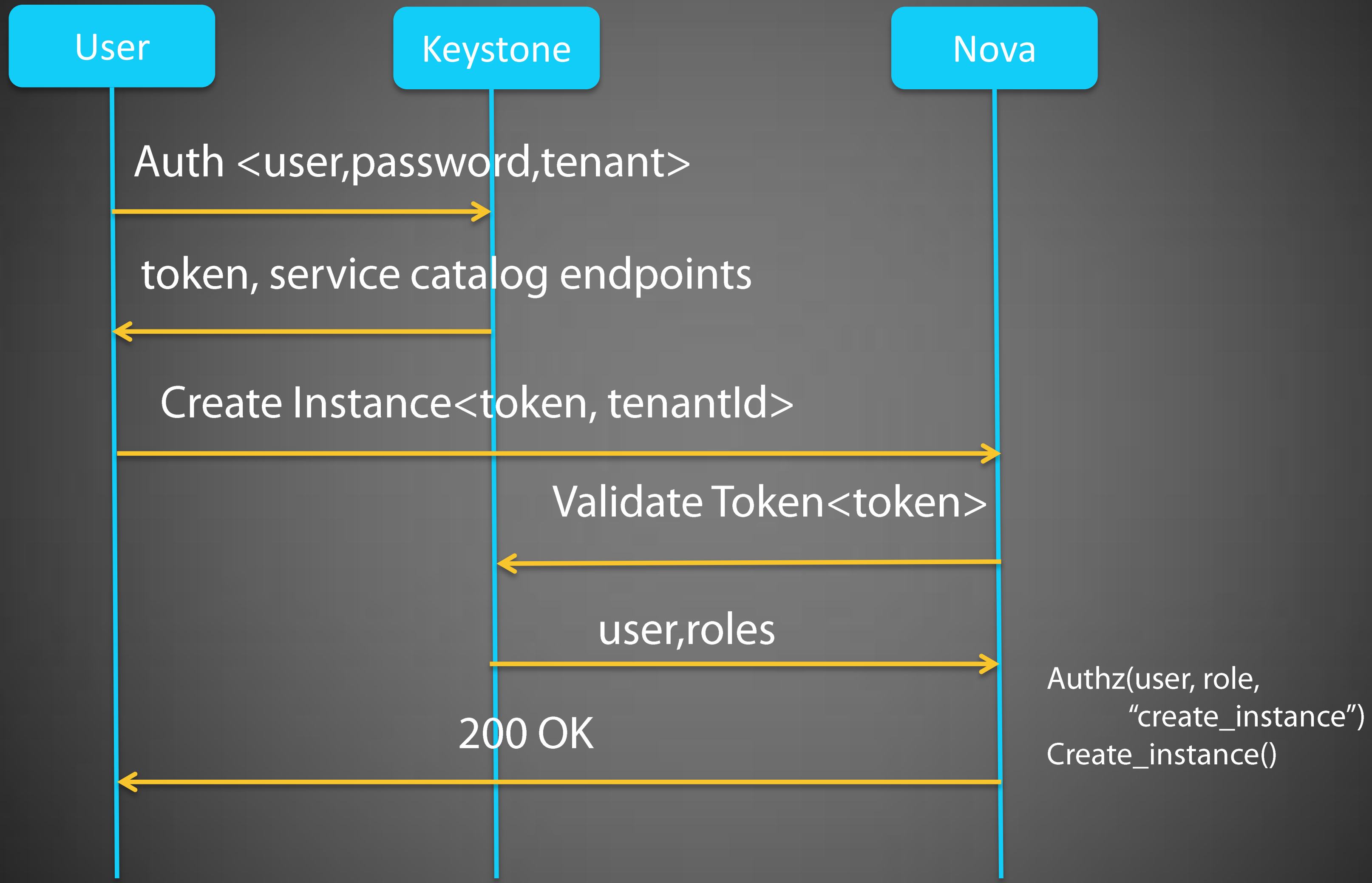
OpenStack Components



Services provided by Keystone

- Identity – Management of Users, Tenants and Roles and validating credentials.
- Token – Creates and validates token.
- Catalog – Provides endpoint registry for services offered by the cloud provider.
- Policy – Rule based authorization engine.

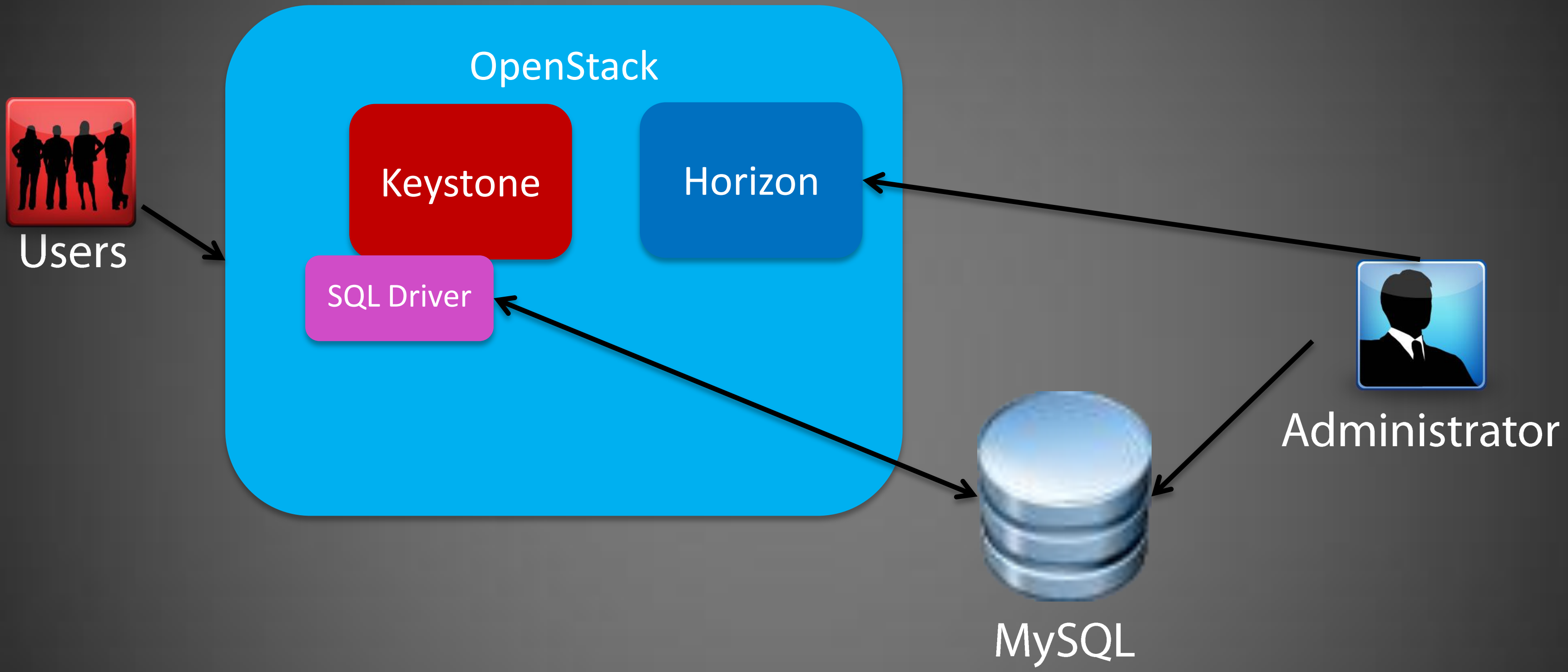
Example: Create Instance


User
Keystone
Nova
Auth <user,password,tenant>
token, service catalog endpoints
Create Instance<token, tenantId>
Validate Token<token>
user,roles
200 OK
Authz(user, role, "create_instance")
Create_instance()

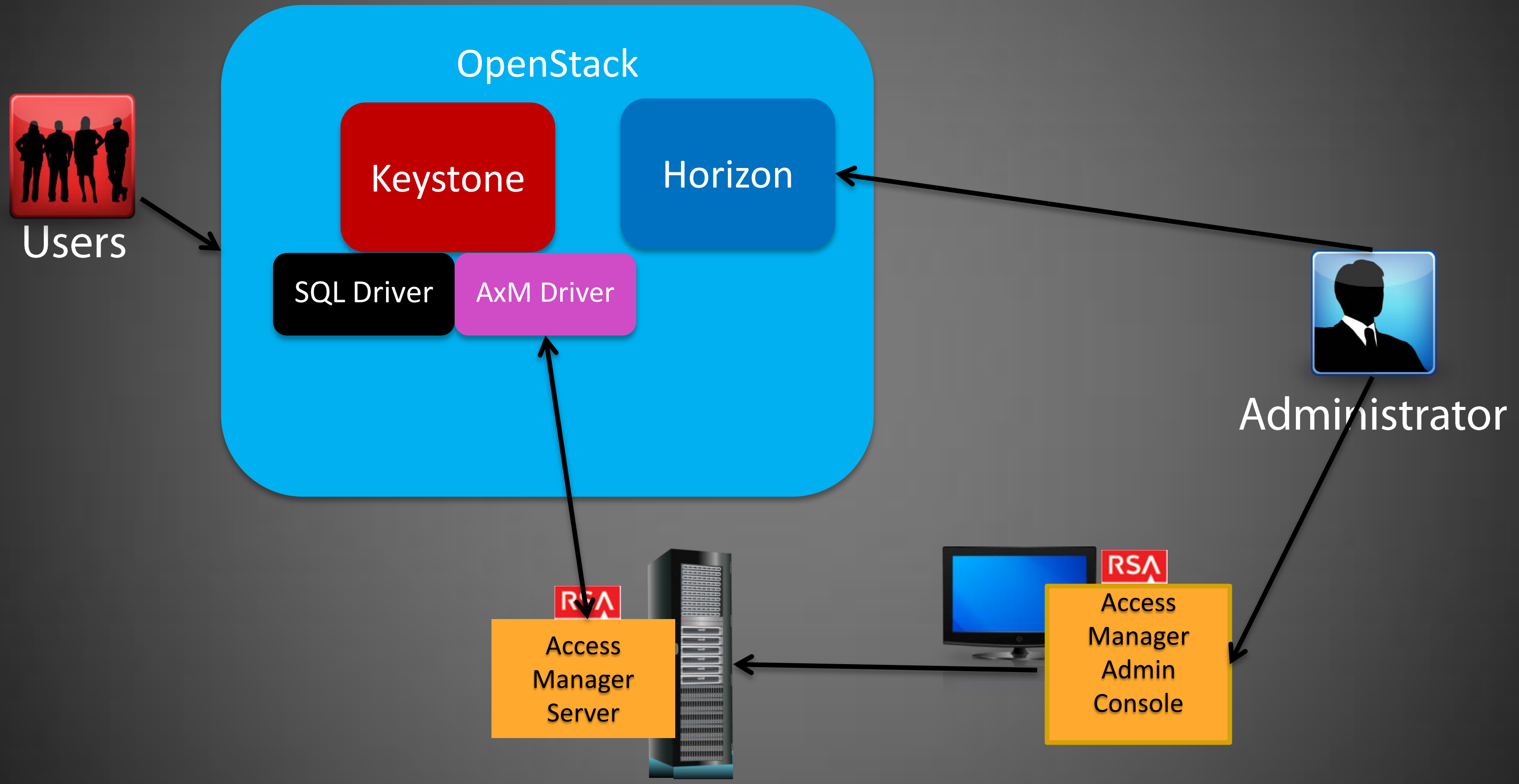
RSA Access Manager

- Web Single Sign On (SSO) product
- Supports consistent controls across applications
- Multiple authentication methods
- Granular authorization
- Agent integration with multiple platforms
- Scales to millions of users

Keystone SQL Driver



Keystone Access Manager Driver



Driver Implementation

```
class Identity(identity.Driver):
    def __init__(self):
        super(Identity, self).__init__()

    # Identity interface
    def get_tenants(self):
      tenants = axm_soap_api.get_all_tenants()
      tenants_ref = []
      for tenant_id in tenants:
        tenants_ref.append(self.get_tenant(tenant_id))
        #return tenants
        return tenants_ref
```

Configuring Keystone (keystone.conf)

```
[identity]
 driver = keystone.identity.backends.sql.Identity
[sql]
connection = mysql://root:password@localhost/keystone?charset=utf8
# the timeout before idle sql connections are reaped
idle_timeout = 200

[identity]
driver = keystone.identity.backends.axm.Identity
[axm]
url = http://10.23.637.562:8080/axm-ws-admin-api/services/AdminAPI?wsdl
adminId =  admin
adminCredentials = admin123
backend_entities = ['UserRoleAssociation', 'Endpoints', 'Role', 'Tenant','User', 'Credentials',
      'EndpointTemplates', 'Token','Service']
```

Integration Challenges

Challenges	Solutions
Access Manager didn’t support “tenants”	Overloaded use of groups

Integration Challenges

Challenges	Solutions
Access Manager didn’t support “tenants”	Overloaded use of groups
No exposed unique IDs for groups, roles, tenants	Locally generated by backend, overloaded Access Manager group and role names

Integration Challenges

Challenges	Solutions
Access Manager didn’t support “tenants”	Overloaded use of groups
No exposed unique IDs for groups, roles, tenants	Locally generated by backend, overloaded Access Manager group and role names
Tokens/sessions are responsibility of Keystone backend	Leveraged MySQL implementation

Integration Challenges

Challenges	Solutions
Access Manager didn't support "tenants"	Overloaded use of groups
No exposed unique IDs for groups, roles, tenants	Locally generated by backend, overloaded Access Manager group and role names
Tokens/sessions are responsibility of Keystone backend	Leveraged MySQL implementation
Username/password only API	Followed common model of overloading password for two factor authentication. No solution for challenge/response, etc.

Integration Challenges

Challenges	Solutions
Access Manager didn't support "tenants"	Overloaded use of groups
No exposed unique IDs for groups, roles, tenants	Locally generated by backend, overloaded Access Manager group and role names
Tokens/sessions are responsibility of Keystone backend	Leveraged MySQL implementation
Username/password only API	Followed common model of overloading password for two factor authentication. No solution for challenge/response, etc.
Significant changes between versions	Manual modifications to upgrade from Diablo to Essex2

Lessons Learned

- Tackle authentication services once
  - We implemented passwords first and then two factor auth
- Stick with one OpenStack version per integration effort
- Implement OpenStack token validation
- When available in OpenStack, use Federated Identity
  - Allows Keystone to focus on OpenStack roles and entitlements
- Improve Role Based Access Control
  - Each OpenStack module currently defines rights independently
  - Hard to manage (policy.conf), hard to reconcile
  - Keystone should manage role->right mapping

Thank You

Suresh Kumar
EMC Solution Group, EMC Corporation
suresh.kumar4@emc.com

Robert Polansky
RSA, The Security Division of EMC
robert.polansky@rsa.com