* **Objective:**

Identify and validate the solutions for component versioning for Lumen-Terrain application

All the components will be versioned using the standard versioning format [Major].[Minor].[Patch].[Build\_Number]

* + 1. MAJOR version when you make incompatible API changes,
    2. MINOR version when you add functionality in a backwards compatible manner, and
    3. PATCH version when you make backwards compatible bug fixes.
    4. BUILD\_NUMBER whenever one of the above version numbers is changed, always reset the build number to 0.

**General Rules for versioning:**

1. A normal version number MUST take the form X.Y.Z where X, Y, and Z are non-negative integers, and MUST NOT contain leading zeroes. X is the major version, Y is the minor version, and Z is the patch version. Each element MUST increase numerically
2. Major version zero (0.y.z) is for initial development. Anything MAY change at any time. The public API SHOULD NOT be considered stable.  
   Major version X (X.y.z | X > 0) MUST be incremented if any backwards incompatible changes are introduced to the public API. It MAY also include minor and patch level changes. Patch and minor version MUST be reset to 0 when major version is incremented.
3. A pre-release version MAY be denoted by appending a hyphen and a series of dot separated identifiers immediately following the patch version. Identifiers MUST comprise only ASCII alphanumeric and hyphen [0-9A-Za-z-]. Identifiers MUST NOT be empty. Numeric identifiers MUST NOT include leading zeroes. Pre-release versions have a lower precedence than the associated normal version. A pre-release version indicates that the version is unstable and might not satisfy the intended compatibility requirements as denoted by its associated normal version. Examples: 1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7, 1.0.0-x.7.z.92.
4. Minor version Y (x.Y.z | x > 0) MUST be incremented if new, backwards compatible functionality is introduced to the public API. It MUST be incremented if any public API functionality is marked as deprecated. It MAY be incremented if substantial new functionality or improvements are introduced within the private code. It MAY include patch level changes. Patch version MUST be reset to 0 when minor version is incremented.
5. Patch version Z (x.y.Z | x > 0) MUST be incremented if only backwards compatible bug fixes are introduced. A bug fix is defined as an internal change that fixes incorrect behavior. Increment the patch level appropriately based on criticality of a bug fix or based on the maintenance release schedule.
6. Increment the build number only as part of an automated test build process. Whenever one of the above version numbers is changed, always reset the build number to 0.

Source: [Semantic Versioning 2.0.0 Specification (SemVer)](https://semver.org/)

**(Option 1):**

**Store the versions of individual components in database:**

Will have a table to store the version information of individual components in application database. Below is the table structure for the same:

Schema diagram

* **Pros:**

a. Central repository to store the version information each component

b. Since RDS is backed up automatically if the system crashes the version information will be archived

c. Since the version information is archived it will be easy to restore the previous state of the application

* **Cons:**

a. All components depend on database for their version information

b. To get the version information for restoration dev-ops will have to get the information using CLI and then re-deploy the appropriate versions from the Artifactory

c. Database update will be required for any component that is being updated. This will not allow the team to independently deploy the components

* **(Option 2)**

**Component is responsible for its own version:**

Each component will maintain its own version and the compatible version of it's dependent component

* **Pros:**
  1. Version information will be stored in distributed location
  2. Each application component will have their individual version information
  3. Any component can be updated independently of one another
  4. No need to know the version of another component
* **Cons:**
  1. Components should know the compatible version of another component
  2. API needs to create to know another component version