

Got it 👍 You're on Angular 7 + AGM Core (Angular Google Maps) with Node 14 + npm 6, and you want to upgrade to Angular 11.

Upgrading across multiple major versions requires sequential migrations, because Angular's CLI and framework enforce version-to-version compatibility.

- ◆ Step 1: Prepare Environment

Node.js: Angular 11 supports Node 10.13, 12.11, or later (Node 14 is fine).

NPM: v6 is fine.

Check:

```
node -v  
npm -v
```

- ◆ Step 2: Clean & Backup

Commit your code to Git (or backup).

Remove node_modules and package-lock.json to avoid dependency conflicts:

```
rm -rf node_modules package-lock.json
```

- ◆ Step 3: Sequential Upgrade

You cannot jump from v7 → v11 directly. Do it step by step using Angular CLI:

3.1 Upgrade Angular CLI first

Globally and locally:

```
npm install -g @angular/cli@7
npm install @angular/cli@7
```

Then run:

```
ng update @angular/cli@7 @angular/core@7
```

Now start upgrading version by version:

➡ v7 → v8

```
ng update @angular/cli@8 @angular/core@8
```

➡ v8 → v9

```
ng update @angular/cli@9 @angular/core@9
```

(Angular 9 introduced Ivy as default. You may need to test your app carefully here.)

➡ v9 → v10

```
ng update @angular/cli@10 @angular/core@10
```

➡ v10 → v11

```
ng update @angular/cli@11 @angular/core@11
```

◆ Step 4: Third-Party Libraries

AGM Core (@agm/core)

Official AGM stopped active support around Angular 8–9.

For Angular 11+, you may face compatibility issues.

Option A: Use @agm/core fork that supports Angular 11.

Option B (Recommended): Switch to @angular/google-maps (official Angular Maps package introduced in Angular 9+).

Example (if staying with AGM):

```
npm install @agm/core@3.0.0-beta.0
```

That version works with Angular 11.

◆ Step 5: Fix Breaking Changes

Check the Angular Update Guide (<https://update.angular.io/?v=7.0-11.0>) for specific breaking changes.

Run:

```
ng lint
ng build
ng serve
```

and fix errors step by step.

◆ Step 6: Verify & Test

Unit tests (ng test).

E2E tests (ng e2e) if you use Protractor.

Update TypeScript to v4.x (Angular 11 requires TS 4.0+).

✅ After these steps, you'll be running Angular 11 successfully.

Do you want me to also compare AGM Core vs Angular's official Google Maps package, so you can decide whether to migrate maps during this Angular upgrade or later?

Great question 👍 — since you're already planning a big Angular upgrade, it's smart to evaluate whether you should also move away from AGM Core (which is essentially unmaintained) to the official `@angular/google-maps` package.

Here's a breakdown of what the conversion looks like:

◆ 1. Installation

Instead of installing `@agm/core`, you add:

```
ng add @angular/google-maps
```

This pulls in the official Angular wrapper for Google Maps.

◆ 2. Module Import

AGM (old way):

```
@NgModule({
  imports: [
    AgmCoreModule.forRoot({
      apiKey: 'YOUR_KEY'
    })
  ]
})
export class AppModule {}
```

`@angular/google-maps` (new way):

```
@NgModule({
```

```
imports: [  
  GoogleMapsModule  
]  
})  
export class AppModule {}
```

👉 The API key is no longer set here; instead you load it in your index.html with the script tag:

```
<script src="https://maps.googleapis.com/maps/api/js?key=YOUR_KEY"></script>
```

♦ 3. Template Changes

This is where most of the work is:

AGM example:

```
<agm-map [latitude]="lat" [longitude]="lng" [zoom]="15">  
  <agm-marker [latitude]="lat" [longitude]="lng"></agm-marker>  
</agm-map>
```

@angular/google-maps equivalent:

```
<google-map height="400px" width="100%" [center]="center" [zoom]="15">  
  <map-marker [position]="center"></map-marker>  
</google-map>
```

👉 Difference:

AGM uses latitude/longitude inputs.

Google Maps package uses center and position objects:

```
center: google.maps.LatLngLiteral = {lat: 37.421995, lng: -122.084092};
```

♦ 4. Features Comparison

✅ Markers, Polylines, Polygons, Circles, InfoWindows → Supported.

✓ Events (mapClick, markerClick, etc.) → Supported but slightly different names.

✓ Performance → Better, since it's maintained and optimized for Ivy.

✗ Some advanced AGM directives (e.g., heatmaps, drawing tools) → You'll need to use the raw Maps JavaScript API or community wrappers.

◆ 5. Migration Difficulty

Simple maps with markers → Easy (mostly template refactor, 1–2 hours work).

Complex maps with many overlays → Medium effort (2–5 days depending on features).

Custom directives from AGM → You'll likely need to use raw Google Maps API inside Angular (TypeScript wrappers).

◆ 6. Long-Term Benefit

AGM Core is unmaintained and may break in future Angular versions.

@angular/google-maps is officially supported by Angular team, so it'll stay up-to-date with Angular releases.

👉 In short:

If you're only using basic markers & maps, migration is straightforward.

If you're using advanced AGM features, you'll need to rewrite some code to use native Google Maps JS API along with the official Angular wrapper.

Do you want me to create a side-by-side code mapping table (AGM directive → Angular Google Maps equivalent), so you can estimate how much refactor your app will need?

Perfect — thanks for clarifying 🙌

You're using @agm/core + @agm/js-marker-cluster + @agm/snazzy-info-window.

Let's evaluate each one against @angular/google-maps so you know what's easy vs. what's tricky:

◆ 1. Base Maps (@agm/core)

✅ Easy migration → Just replace <agm-map> with <google-map> and convert [latitude]/[longitude] to [center].

Example:

AGM:

```
<agm-map [latitude]="lat" [longitude]="lng" [zoom]="12">
  <agm-marker [latitude]="lat" [longitude]="lng"></agm-marker>
</agm-map>
```

Google Maps:

```
<google-map height="400px" width="100%" [center]="center" [zoom]="12">
  <map-marker [position]="center"></map-marker>
</google-map>
```

◆ 2. Marker Clustering (@agm/js-marker-cluster)

⚠️ Harder migration.

The official Angular Maps package does not include clustering out of the box. But Google provides an official MarkerClusterer utility in the Maps JavaScript API. You can integrate it manually in Angular:

Steps:

1. Install clusterer package:

```
npm install @googlemaps/markerclusterer
```

2. In your component:

```
import { MarkerClusterer } from "@googlemaps/markerclusterer";
```

```
@ViewChild(GoogleMap) map!: GoogleMap;
```

```
ngAfterViewInit() {  
  const markers = this.locations.map(  
    (loc) => new google.maps.Marker({ position: loc })  
  );  
  
  new MarkerClusterer({ markers, map: this.map.googleMap! });  
}
```

👉 Migration effort: medium — you'll need to write a little wrapper, but once done, it works cleanly.

♦ 3. Snazzy Info Window (@agm/snazzy-info-window)

⚠ Tricky migration.

@agm/snazzy-info-window is a wrapper around the Snazzy Info Window JS library, which provides fancy styled info windows.

The official Angular Maps package only has the standard <map-info-window>.

If you need custom styling:

Option A (Easy): Use `<map-info-window>` and style its content with Angular templates.

Option B (Harder): Continue using `snazzy-info-window` directly as a JS library (you'd integrate it manually with Angular, similar to how you did with AGM).

Example official way:

```
<map-marker #marker="mapMarker" [position]="center" (mapClick)="openInfo(marker)">
</map-marker>
```

```
<map-info-window>{{ infoContent }}</map-info-window>
```

```
@ViewChild(MapInfoWindow) infoWindow!: MapInfoWindow;
```

```
openInfo(marker: MapMarker) {
  this.infoWindow.open(marker);
}
```

◆ Migration Difficulty Summary

Feature	AGM	Angular Google Maps Equivalent	Migration Difficulty
Base Map	agm-map	google-map	✓ Easy
Markers	agm-marker	map-marker	✓ Easy
Clustering	agm/js-marker-cluster	Use @googlemaps/markerclusterer	⚠ Medium
Snazzy Info Window	agm/snazzy-info-window	map-info-window (or custom styling)	⚠ Medium–Hard

◆ Recommendation

If you're doing the Angular 7 → 11 upgrade anyway, this is a good time to migrate, because AGM is already behind and not maintained.

Base maps + markers → Quick wins (1–2 days).

Clustering & Snazzy Info Windows → Expect some manual work, especially styling. (~2–4 more days depending on customization).

Would you like me to draft a migration guide with code snippets for each AGM feature you're using (map, marker cluster, snazzy info window) so you can do a gradual switch rather than rewriting all at once?

Got it 👍 Let's build a side-by-side migration table comparing AGM (@agm/core, @agm/js-marker-cluster, @agm/snazzy-info-window) with @angular/google-maps equivalents.

AGM → Angular Google Maps Migration Table

Feature	AGM (@agm/*)	Angular Google Maps (@angular/google-maps)	Notes / Migration Tips
---------	--------------	--	------------------------

Map	<code>html
<agm-map [latitude]="lat" [longitude]="lng" [zoom]="12"></agm-map>
html
<google-map [center]="center" [zoom]="12" height="400px" width="100%"></google-map>

ts
center: google.maps.LatLngLiteral = {lat: 37.42, lng: -122.08};</code>		AGM uses latitude / longitude. Official maps use a center object (LatLngLiteral).
-----	---	--	---

Marker	<code>html
<agm-marker [latitude]="lat" [longitude]="lng"></agm-marker>
html
<map-marker [position]="center"></map-marker>
</code>		AGM splits lat/lng, official maps uses a single [position] object.
--------	--	--	--

Marker Click Event	<code>html
<agm-marker (markerClick)="onMarkerClick()"></agm-marker>
html
<map-marker (mapClick)="onMarkerClick()"></map-marker>
</code>		Event names differ slightly.
--------------------	--	--	------------------------------

Marker Cluster	<code>html
<agm-map><agm-marker-cluster><agm-marker *ngFor="let m of markers" [latitude]="m.lat" [longitude]="m.lng"></agm-marker></agm-marker-cluster></agm-map>
</code>	<code>ts
import { MarkerClusterer } from "@googlemaps/markerclusterer";


@ViewChild(GoogleMap) map!: GoogleMap;

ngAfterViewInit() {
 const markers = this.locations.map(loc => new google.maps.Marker({ position: loc }));
 new MarkerClusterer({ markers, map: this.map.googleMap! });
}

</code>	No <map-marker-cluster> directive. You manually create a MarkerClusterer instance from Google's official utility.
----------------	---	--	---

Info Window (Default) `html
<agm-map><agm-marker [latitude]="lat" [longitude]="lng"></agm-marker><agm-info-window>Info content</agm-info-window></agm-map>
html
<google-map><map-marker #marker="mapMarker" [position]="center" (mapClick)="openInfo(marker)"></map-marker><map-info-window>Info content</map-info-window></google-map>

ts
@ViewChild(MapInfoWindow) infoWindow!: MapInfoWindow;
openInfo(marker: MapMarker) { this.infoWindow.open(marker); }` Very similar, just slightly different syntax.

Snazzy Info Window `html
<agm-snazzy-info-window [isOpen]="true"><ng-template><h3>Custom styled!</h3></ng-template></agm-snazzy-info-window>
<map-info-window>` with Angular templates + CSS for styling. For advanced “snazzy” looks, you’d integrate the snazzy-info-window JS library manually. Migration effort depends on how custom your snazzy windows are.  No direct equivalent. Use

♦ Key Takeaways

Base maps & markers → trivial changes (rename tags + adjust inputs).

Marker clustering → manual setup with `@googlemaps/markerclusterer`.

Snazzy Info Window → redesign using `<map-info-window>` + Angular templates (or continue using the external JS library).

Would you like me to prepare a full working Angular 11 component example that includes:

`<google-map>`

markers (looped from an array)

clustering with `MarkerClusterer`

and a styled info window (snazzy-like)

so you can test it right after migrating?

