Workflow:

Phase 1: The Minimal Viable Simulation (MVS)

Before moving to complex geometries, you must establish a baseline to ensure the LBM parameters (relaxation time, lattice weights) are physically consistent.

- Lattice Initialization: Setup the D2Q9 (2D) lattice using LBMpy with a basic fluid domain.
- Boundary Conditions: Implement the "Slip" vs. "No-Slip" bottom boundary logic to represent the presence or absence of the track.
- Validation: Test your code against a standard Poiseuille flow or Lid-driven cavity flow. If possible, compare these initial runs to analytical solutions to ensure the viscosity is being handled correctly.
- Verification: Test Reynolds number sweep on simple geometries (cylinder or flat plate) before introducing the F1 wing (verify turbulence transition detection methods work before complexity increases).

Phase 2: Dynamic Reynolds & Geometry Integration

Once the "virtual wind tunnel" is stable, you can introduce the complexity of the F1 front wing and variable flow parameters.

- Proxy Implementation: Introduce the geometric proxy representing the F1 front wing into the domain.
- Parameter Sweeping: Develop a script to systematically vary the Reynolds number (Re) and ride height. This is crucial for identifying the "tipping point" or subcritical bifurcation you mentioned in your hypothesis.
- Begin creating first/basic visualisations.
- Data Management: Ensure your script automatically exports flow fields at specific intervals to your Git-tracked data management system.
    - Automated naming conventions for simulation runs (e.g. `Re{value}_h{height}_slip{Y/N}_timestamp`)
    - A metadata logging system that captures all parameters for each run.

Phase 3: Transition & Stability Analysis

This is the core of your complex systems investigation—identifying the nature of the phase transition.

- Bifurcation Tracking: Implement tools to measure the fluctuations in the velocity field (turbulent kinetic energy (TKE) calculation, track temporal evolution of velocity variance, autocorrelation functions to distinguish chaotic vs. periodic behavior)
- A sudden increase in the variance of velocity components often signals the transition from laminar to turbulent flow.

- Force Calculations: Add functions to calculate the downwards force (lift/downforce) generated by the wing proxy at different pitch angles.
- Sensitivity Testing: Run simulations with minute perturbations in initial conditions to test the "sensitivity to initial conditions" typical of chaotic systems.
- Define Explicit Convergence Criteria: what constitutes a "steady state" for laminar flow. Set statistical convergence criteria for turbulent flow (e.g., time-averaged quantities stabilize).

Phase 4: Scaling and Visualization

If the 2D simulations become too computationally expensive or if you move to higher-resolution grids, you will need to pivot to high-performance computing.

- Snellius Migration: If the local Python environment lags, migrate the LBMpy kernels to Snellius to utilize GPU/parallel acceleration.
- Visualization Pipeline: Develop a script to generate animations of the "vortex shedding" and turbulent structures, as these are the emergent properties of your microscopic model.